Commodore INFO

PRIJS ƒ 7.95 / Bfr. 160

8 710966 001332

ONAFHANKELIJK BLAD VOOR COMMODORE GEBRUIKERS

Jaargang 7, NO.1, febr. /mrt. '90

LISTINGS:

- Shiftgame (C-64)
- Paard (C-64)
- Kaartspel (C-128)
- Alert (Amiga)

DE POWERCARTRIDGE
ON-BEVEILIGEN VAN BASIC
CLS IN BASIC
VES ONE VIDEOSTUDIO
AMIGA RAM UITBREIDEN

Vaste rubrieken:

- Tips en Trucs voor de C-64
- Amiga Tips
- Graphics op de 64
- GEOS Machinetaal

Commodore Info

Verschijnt 8x per jaar
Jaarg.7, no.1, febr/mrt. 1990

Uitgave:
Sala Communications

Uitgever:
V. Sharfman

Redactie:
ir. L. Sala hoofdredacteur
J. Bodzinga adj. hoofdred.
drs. J. Boers eindredacteur
H. Smeenk, drs. U. Schuurmans, W. Scheer R. Goudriaan, B. Munniksma, B. Venema, P. Boncz, MGCC/Johan & Johan

Produktie
drs. H. Zoete
J. Broekhuizen, M. van Zijl

Redactiesecretariaat:
R. van Zalingen

Strip:Bert Tier
Illustraties: Ben van Mierlo

Advertentie-exploitatie:
Ing. V. Sala, Ing. B. Sala, D. van Vlijmen
Postbus 43048, 1009 ZA Amsterdam
tel. 020-273198

Redactie adres:
Postbus 43048, 1009 ZA Amsterdam
tel. 020-228871

Listingtelefoon: (ma: 17.00-21.00 u)
02155-25162

Abonnementen en administratie:
Sandra v.d. Meyden en Marjo Jansen
Postbus 43048
1009 ZA Amsterdam
tel. 020-248006

Vragen betreffende abonnementen ontvangen wij bij voorkeur schriftelijk, met meesturen van het omslagetiket.

Abonnement:
Voor 8 nummers *f* 47,50 of Bfr. 975 per jaar. Betaling op giro 4985259 (België: BBL nr. 310050602562) t.n.v. SAC/Commodore-Info. Oude nummers kunt U alleen krijgen bij vooruitbetaling van *f* 6,75 op de bovenstaande rekening. Ook telefonische opgave voor een abonnement is mogelijk. Bel GRATIS 06-02242222 (teleservice), elke dag tot 20.20 uur (dus ook in het weekend). België: 115555, dagelijks tot 22.00 uur. Deze telefoonnummers zijn alleen bedoeld voor opgave van NIEUWE abonnementen.
Opzegging dient schriftelijk te geschieden uiterlijk twee maanden voor de aanvang van een nieuwe abonnementsperiode van een jaar.

Zetwerk & druk: NDB, Zoeterwoude

Distributie:
In Nederland: Betapress, Gilze
In België: AMP, Brussel

ISSN: 0169-3085

Inhoud van dit nummer

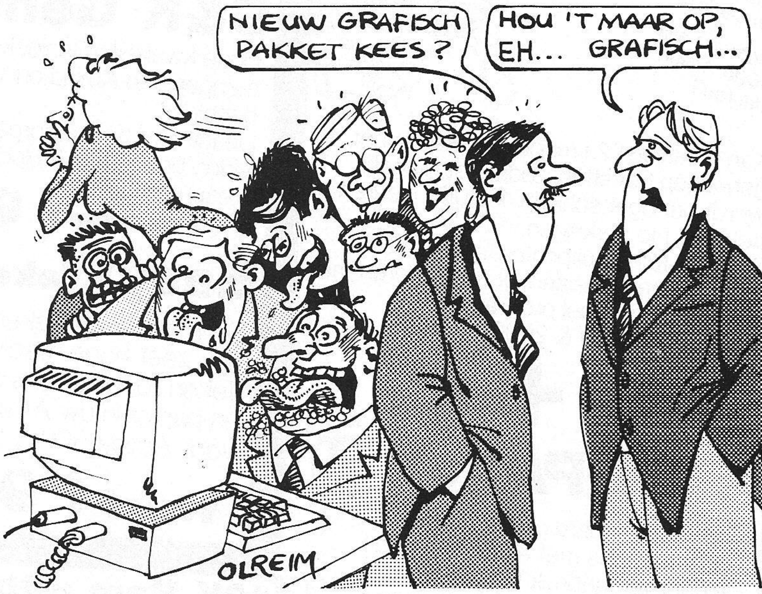
Redactioneel

Commodore heeft met de aankondiging van nieuwe Amiga systemen zowel in de huiselijke als zakelijke sfeer een belangrijke stap gedaan. De Amiga als kern van een geïntegreerd audio-visueel concept met CD-ROM en in de toekomst CD-interactive zal veel mensen aanspreken. Mij in ieder geval. Thuis moet je in staat zijn om alle systemen samen te laten werken, zodat 'dubbele' hardware zoveel mogelijk wordt voorkomen. Een computer kan prima zorgen voor het tunen van radio en TV, programmeren van de video en opslaan van gedigitaliseerde audio- of video. Voor het gebruik van de computer thuis, afgezien van het naar thuis verlegde werk, is zo'n geïntegreerde aanpak voor mij persoonlijk ook een vereiste. De techniek is nu zover dat dit ideaal verwezenlijkt kan worden. De (huis)computer kan dan de plaats in de huiskamer krijgen die hij/zij als pure hobby- of werkcomputer niet altijd verdient. Commodore kon hierin wel eens een belangrijke stap gaan doen.
Ook met de Amiga 2500 lijkt Commodore goed bezig, vooral in de professionele markt, alhoewel het verkooprijp maken van deze systemen zowel met Amiga als met Unix besturingssysteem wel erg lang op zich heeft laten wachten. Met de nog niet officieel gepresenteerde 3000 zegt Commodore een geheel nieuwe Amiga te hebben ontworpen, en we zijn dan ook erg benieuwd wat daar uit gaat komen. Laten we hopen dat Commodore dit systeem feitelijk aankondigt wanneer het ook leverbaar is, dan kunnen we er direct mee aan de slag.
Jan Boers

Graphics op de 64

Deel 7: Bank switching & Scrolling

Hylke Sprangers en Michel de Boer gaan fris het nieuwe jaar in met de zevende aflevering van de cursus "Graphics op de 64". De vorige aflevering over ComputerKunst was een luchtig intermezzo dat de mogelijkheden van de Commodore 64 op het grafische gebied aanstipte. In deze aflevering wordt nog een keer teruggekeerd naar de theorie om vervolgens in de volgende afleveringen genadeloos uit te pakken op het gebied van flitsende animaties, special effects en ComputerKunst.



In deze aflevering zullen we het gaan hebben over bank switching en scrolling. Het onderwerp bank switching heeft veel raakvlakken met de theorie die in de voorgaande afleveringen ter sprake is gekomen. We hebben in het begin van de cursus onderscheid gemaakt tussen drie grafische objecten, te weten: karakters, sprites en high resolution. Het onderwerp bank switching heeft te maken met de verschillende geheugengebieden, waar deze drie objecten kunnen worden opgeslagen. Met het onderwerp scrolling maken we een begin met animatie.

Video banks

In de inleiding hebben we het al gezegd, dat we wederom de drie grafische objecten onderscheiden De besturing van deze grafische objecten, de karakters, de sprites en hires, wordt gedaan door de VIC (Video Interface Chip). De besturing van de objecten moet gedaan worden door verschillende waarden in bepaalde geheugen adressen te poken. Deze adressen bevinden zich van 53248 tot 53294.
De Commodore 64 heeft 65536 geheugen adressen, oftewel 64 K. Om 64 K te kunnen adresseren zijn 16 bits nodig, immers 2~16 = 65536. De VIC heeft echter maar 14 adreslijnen. Dit betekent dat deze maar 14 bits kan behandelen. Met deze 14 bits kunnen 2~14 = 16384 bytes worden geadresseerd; dit is 16 K. De 64 K van de Commodore 64 is derhalve in 4 secties verdeeld van elk 16 K. Uit elke van de 4 secties kan de VIC de benodigde gegevens halen, met de beperking dat de VIC per keer maar toegang heeft tot 1 sectie. De gegevens waar we nu over spreken zijn de gegevens van de drie grafische objecten. Bijvoorbeeld waar het scherm staat, waar de karakterset zich bevindt enz.
Deze 4 secties worden ook wel video banks genoemd. Zoals gezegd kan de VIC maar 1 bank tegelijk overzien. Als we naar een andere bank willen, moeten we dit doorgeven aan de computer. Het veranderen van bank wordt bank switching genoemd. In figuur 1 staat de geheugen opdeling in de 4 banks. U ziet dat bank 0 van adres 0 tot adres 16383 loopt. Dit is de bank waar u standaard in werkt. Ga maar na, het schermgeheugen begint op adres 1024, en ligt dus in bank 0. Ook voor het kiezen van het gebied voor een sprite bent u veroordeeld tot de adressen in bank 0.

Wisselen van bank

We gaan nu bekijken hoe we kunnen wisselen van de ene bank naar de andere, het echte bank switching dus. Om te beginnen moeten de eerste twee bits van adres 56578 aan worden gezet:

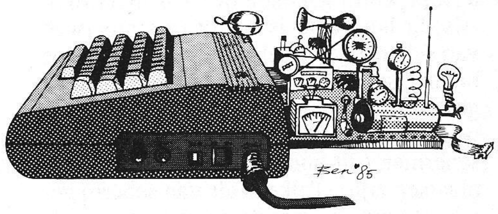
```
POKE 56578, PEEK(56578) OR 3
```

Als dit gedaan is, zijn de poorten PA0 en PA1 van de CIA 2 (Complex Interface Adapter) uitgangspoorten geworden. Dit betekent niets anders dan dat we nu aan de computer door kunnen geven in welke bank we willen werken. Voor het kiezen van een van de vier video banks hebben we twee bits nodig. Deze twee bits zijn de eerste twee bits van adres 56576. Om een bank te selecteren moet u de volgende regel in typen:

```
POKE 56576, PEEK(56576) AND
      252 OR A
```

De "AND 252" is nodig om eerst de eerste twee bits uit te zetten. Vervolgens moet u voor A de waarde invullen die in figuur 1 staat vermeld. Voor bank 0 heeft A bijvoorbeeld de waarde 3. Met de nu verkregen kennis kunt u wel van bank switchen, maar er zal dan niet zoveel gebeuren. De computer zal hoogstens flink vast slaan. Dit komt omdat u in de nieuw gekozen bank ook nog aan de computer

moet doorgeven waar de grafische objecten staan. Hoe dit moet vertellen we in de volgende paragrafen.
In het kort samengevat: De Commodore 64 kent 4 banks. U kunt zelf willekeurig een bank kiezen om in te werken. Dit betekent dat in de gekozen bank alle gegevens van de grafische objecten staan of komen te staan. We hebben het dan bijvoorbeeld over de plaats van het scherm, de plaatsen van sprites enz. Met behulp van de adressen 56576 en 56578 kan van bank worden gewisseld.



Het karakter geheugen

In de eerste aflevering van deze cursus hebben we het gehad over de karakters. U kunt zelf de karakterset herdefiniëren. De Commodore 64 kent 256 verschillende karakters. Elk karakter is opgebouwd uit 8 bytes. Er zijn dus 8 * 256 = 2048 bytes nodig om een karakterset in op te slaan. Standaard heeft de Commodore twee verschillende karaktersets, te weten de lower- en upper-case karakterset. Om deze twee karaktersets op te slaan, is 4 K geheugen nodig. De eerste karakterset begint op adres 53248 en de tweede op 55296. Omdat dit gebied normaal door de VIC gebruikt wordt, moet eerst het video geheugen worden uitgeschakeld voordat u bij de twee karaktersets kunt. Dit kan gedaan worden door:

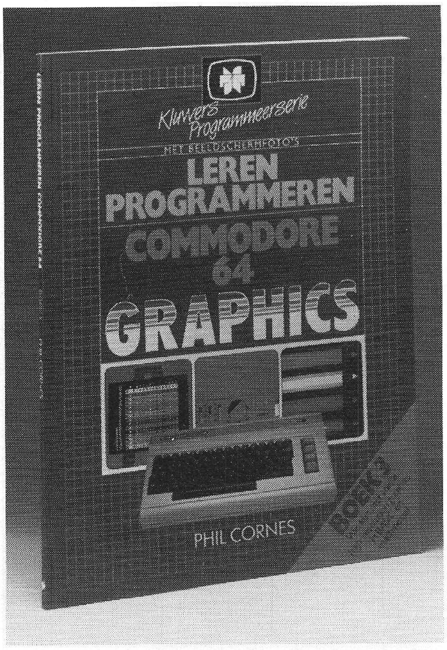
```
POKE 56334, PEEK(56334) AND
     254:
POKE 1, PEEK(1) AND 251
```

U kunt dan de twee karaktersets uitlezen en eventueel kopiëren. Het video geheugen kan weer worden aangezet met:

```
POKE 1, PEEK(1) OR 4:
POKE 56334, PEEK(56334) OR 1
```

U kunt zelf ook een karakterset maken. Omdat het opslaan van een karakterset 2 K geheugen kost, gaan er in een bank 16/2 = 8 verschillende sets. Er kunnen maximaal 8 karaktersets in een bank worden opgeslagen. In figuur 2 staan de 8 gebieden voor bank 0. Voor de gebieden in de andere drie banks, moet u telkens het beginadres van de bank optellen bij de gebieden die gegeven staan voor bank 0. Voor het doorgeven van het gebied dat u wilt gebruiken, moet u de bits 1 tot 3 van adres 53272 gebruiken:

```
POKE 53272, PEEK(53272) AND
     240 OR B
```



U kunt de waarde van B opzoeken in figuur 2. Sommige lezers zullen iets vreemds ontdekt hebben in het voorgaande verhaal. We hebben namelijk verteld dat de VIC maar 16 K aan kan, en dat dat standaard de eerste 16 K zijn van de computer. Maar dan zouden in die eerste bank toch ook de 2 karaktersets moeten zijn opgeslagen? Deze staan echter opgeslagen vanaf adres 53248. Hoe zit dat dan? Wel, de makers van de Commodore hebben een slimme truc toegepast. De VIC denkt dat de karaktersets zijn opgeslagen vanaf adres 4096. De VIC beschouwt het geheugengebied van adres 4096 tot 8191 dan ook als karakterROM. De bits 1 tot 3 van adres 53272 wijzen naar het derde gebied (4096 - 6143) in bank 0. Of, als de Commodore 64 in de upper-case staat, naar het vierde gebied in bank 0 (6144 - 8191). In het echt staan de standaard sets opgeslagen vanaf adres 53248. Binair is 53248 %1101 0000 0000 0000. Hakken we hier de eerste twee bits vanaf (VIC heeft maar 14 adreslijnen) dan krijgen we %01 0000 0000 0000. Dit is precies adres 4096, en daarom beschouwt de VIC het gebied van adres 4096 tot adres 8191 als karakterROM.
Nu zult u misschien denken, als de VIC het derde en vierde gebied in bank 0 als karakterROM beschouwt, dan geldt dit waarschijnlijk ook voor het derde en vierde gebied in de andere drie banks. Dit is echter maar ten dele waar. De VIC beschouwt het derde en vierde gebied alleen in bank 0 en in bank 2 als karakterROM. In bank 1 en in bank 3 kent de VIC geen karakterROM, wat tot gevolg heeft dat u in die twee banks geen beschikking heeft over een karakterset. Als u in een van die twee banks gaat werken, moet u zelf een karakterset definiëren. Wat ook kan natuurlijk, is een van de standaard karaktersets kopiëren naar die bank. Want in welke bank u ook werkt, u heeft altijd toegang tot het karakterROM vanaf adres 53248.
We vatten het voorgaande verhaal wederom kort samen. In een bank zijn er acht gebieden waarin een karakterset kan worden gezet. Het uitkiezen van een gebied kan gedaan worden met adres 53272. Normaal werkt u met het derde gebied. Het echte karakterROM is gesitueerd vanaf adres 53248. In bank 0 en in bank 2 wordt het derde en vierde gebied beschouwd als karakterROM.

Rectificatie graphics

In het vorige nummer (NO. 8, dec. '89/ jan. '90) zijn er enkele zetfouten in het artikel "Graphics op de 64" geslopen.

Ten eerste zijn alle groter en kleiner dan tekens ('', '') weggevallen. In de een na laatste alinea van de paragraaf "Lissajous figuren" staat de regel: (a1 + a2) 160 en (a3 + a4) 100. Dit moet zijn (a1 + a2) 160 en (a3 + a4) 100.
In de alinea onder de listing "sinus krommen" staat: (a1 * a2) 160 en (a1 * a3) 100. Dit moet zijn (a1 * a2) 160 en (a1 * a3) 100.

In de listing draaiende lijnen is regel 130 fout. Deze regel moet zijn:
130 if x or x319 or y or y199 then 190
In listing 1 (lissajous slang) moet regel 30 veranderd worden in:
30 if s53010 then print"fout in data!"
Ten tweede zijn alle machtverheffingstekens (~) veranderd in tildes (slangetjes). In de listing "boom van Pythagoras" moeten de tildes worden veranderd in vertikale pijlen.
Excuses voor de gemaakte fouten

Het karakterROM

Het feit dat VIC het derde en vierde gebied in bank 0 en in bank 2 beschouwt als ROM, heeft interessante consequenties voor de drie grafische objecten. In deze in totaal 4 gebieden kunnen namelijk geen grafische objecten staan, om de doodsimpele reden dat in ROM niet geschreven kan worden. Let wel, deze gebieden zijn wel gechikt om gegevens in te zetten, bijvoorbeeld een programma. Dit gebied is immers wel RAM, met de beperking dat VIC het als ROM ziet. In deze gebieden (4096-8191 en 36864-40959) kunnen dus geen grafische objecten staan. Geen sprites, geen zelfgemaakte karakterset, geen scherm enz. Nu is het ook te begrijpen waarom in bank 1 en in bank 3 deze beperking niet geldt. In deze twee banks hebben we dan meer gebieden waarin we grafische objecten kunnen zetten. Het nadeel is dat we in deze twee banks geen standaard karakterset tot onze beschikking hebben.

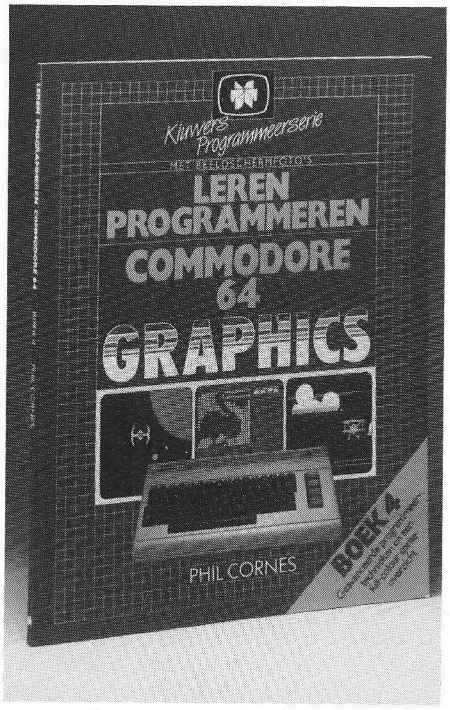
Het schermgeheugen

U kunt zelf bepalen waar het schermgeheugen zich bevindt in de bank. Standaard wordt het scherm neergezet van adres 1024 tot adres 2023 in bank 0. De 1000 bytes van het scherm kunnen op 16 verschillende plaatsen worden neergezet in een bank. In figuur 3 staan de 16 plaatsen voor bank 0. U ziet dat de bank in 16 delen van 1 K is opgesplitst. Op elk beginadres van de 16 delen van 1 K kan het schermgeheugen worden gezet. U kunt bijvoorbeeld voor het laatste geheugengebied kiezen om het scherm neer te zetten. Het scherm zou dan lopen van adres 15360 tot adres 16359. Maar let wel, dit is de situatie voor bank 0. Als u bijvoorbeeld inmiddels was overgestapt naar een andere bank moet u het beginadres van deze bank er nog bij optellen. We geven een voorbeeld. U wilt werken in bank 2. U voert de 2 programma regels uit die we hebben gegeven. Voor A moet u dan de waarde 1 hebben (zie figuur 1):

```
POKE 56578, PEEK(56578) OR 3:
POKE 56576, PEEK(56576) AND
    252 OR 1
```

U zit dan in bank 2. U hebt nu wederom keuze uit 16 verschillende gebieden om het scherm in te zetten. We kiezen nu het derde gebied. Nu loopt het scherm dus van adres 32768+2048 = 34816 tot adres 32768+3047 = 35815. Het beginadres van de bank moet bij het schermgebied opgeteld worden. In ons voorbeeld is dit beginadres 32768 (bank 2).
Nu moet het gebied waar we het scherm willen zetten nog aan de computer doorgegeven worden. Met de hoogste 4 bits van adres 53272 kan aan de computer verteld worden welk schermgebied we willen hebben. Dit kan gebeuren door:

```
POKE 53272, PEEK(53272) AND
    15 OR C
```



De variabele C kunt u opzoeken in figuur 3. Voor het laatste gebied heeft C bijvoorbeeld de waarde 240. Als u deze regel ingevoerd heeft, kunt u al wel karakters op het scherm POKEN. Het PRINTEN van karakters gaat echter nog niet. Eventuele invoer van het toetsenbord wordt dan ook niet op het scherm gezet. Daarvoor moeten we eerst nog de screen-editor van de kernal waarschuwen dat het scherm is verplaatst. Daarvoor delen we eerst het beginadres van het schermgeheugen door 256. Dit getal moet in adres 648 gepoked worden:

```
POKE 648, (begin
    schermgeheugen)/256
```

Normaal staat het schermgeheugen op adres 1024. in adres 648 staat dan 1024/256 = 4. In ons eerdere voorbeeld lieten we het schermgeheugen beginnen op adres 34816. In adres 648 moet dan de waarde 34816/256 = 136 worden gezet.
Verder moet u er rekening mee houden, dat het scherm niet in de gebieden 4096-8191 en 36864-40959 kan worden gezet vanwege het karakterROM. Ook moet u er rekening mee houden, dat als u het scherm verplaatst naar een gebied in bank 1 of in bank 3, dat er geen standaard karakterset meer aanwezig is. Tenzij u zelf een karakterset definieert, ziet u dan een hoop troep op het scherm. Als u van bank 0 overschakeld naar bank 1, zal het scherm ook meegaan naar bank 1. Dat wil zeggen, als het scherm in het tweede gebied stond in bank 0, dan zal het nu in het tweede gebied staan in bank 1.
Er kunnen ook twee schermen bij worden gehouden in dezelfde bank (of in verschillende banks natuurlijk). Het volgende programma geeft hier een voorbeeld van. Door telkens op F1 te drukken kan van scherm gewisseld worden. Het eerste scherm begint op adres 1024 en het tweede op adres 15360. Er hoeft dus niet van bank veranderd te worden. Het programma is in machinetaal geschreven. In de interrupt wordt gekeken of F1 is ingedrukt. Als dit het geval is wordt er simpelweg van scherm gewisseld. Het enige addertje bij het schrijven van een dergelijk programma is de cursor. Er moet onthouden worden waar de kursor staat op beide schermen (dit kunnen twee verschillende plaatsen zijn). Telkens als van scherm gewisseld wordt moet de cursor weer op de oude plaats worden teruggezet.

```
5 rem screen swap
10 for x=0 to 90:read a:i=i+a
20 poke x+49152,a:next
30 if i11219 then print"fout
   in data"
100 data
    120,169,192,141,21,3,169,13
110 data
    141,20,3,88,96,165,197,201
120 data
    4,240,3,76,49,234,173,136
130 data
    2,201,4,240,18,169,4,141
140 data
    136,2,173,24,208,41,15,9
150 data
    16,141,24,208,76,62,192,169
160 data
    60,141,136,2,173,24,208,41
170 data
    15,9,240,141,24,208,162,100
180 data
    32,179,238,202,224,0,208,24
    8
190 data
    164,253,165,211,133,253,24,
    166
200 data
    254,165,214,133,254,32,240,
    255
210 data 76,49,234
```

Het kleurengeheugen

In tegenstelling tot het schermgeheugen kan het kleurengeheugen niet verplaatst worden. Het kleurengeheugen loopt altijd van adres 55296 tot adres 56295. U moet dus oppassen als u twee verschillende schermgeheugens bijhoudt. Deze twee

verschillende schermen, waarop onafhankelijk van elkaar karakters kunnen worden gezet, hebben steeds hetzelfde kleurengeheugen. Iets dergelijks geldt ook voor het Basic RAM. U kunt wel twee verschillende schermen bijhouden, maar de twee schermen delen allebei hetzelfde RAM, tenzij u dit zelf natuurlijk veranderd. Als u op het ene scherm een programma intypt, en vervolgens overschakelt naar een ander scherm, heeft u daar ook toegang tot het programma.

De sprites

In de tweede aflevering van deze cursus hebben we het over de sprites gehad. Een sprite (in het engels betekent sprite zoiets als dwaalgeest) bestaat uit 63 bytes. Deze 63 bytes kunnen niet willekeurig in het geheugen worden gezet. We hebben destijds drie eisen geformuleerd waaraam het geheugengebied moest voldoen:

- Het eerste adres van het geheugengebied moet een veelvoud van 64 zijn;
- Het geheugengebied moet tussen adres 0 en adres 16384 liggen;
- Het gebied tussen de adressen 4096 en 8192 kan niet gebruikt worden;

Nu kunnen we verklaren waar deze eisen vandaan komen. De derde eis komt rechtstreeks voort uit het feit, dat het gebied tussen 4096 en 8192 karakterROM is. De tweede eis is ook duidelijk, daar we standaard in bank 0 werken, en bank 0 loopt van adres 0 tot adres 16383. De eerste eis heeft te maken met de sprite pointers. Door steeds met gebieden van elk 64 bytes te werken, gaan er 16384/64 = 256 gebieden in een bank. Voor elke van de acht sprites is er een sprite pointer, die naar het gebied wijst waar de betreffende sprite zich bevindt. Een sprite pointer is gewoon een adres. In dat adres (een byte) kan precies een waarde van 0 tot 255, lineair corresponderend met een van de 256 gebieden om de sprites in te zetten. De bovenstaande drie eisen gelden alleen in bank 0. Zoals als eerder opgemerkt is er in bank 1 en in bank 3 geen karakterROM. Daar kan een sprite dus in principe in elk van de 256 gebieden. Er moet echter opgepast worden met de sprite pointers. Deze verhuizen namelijk steeds met het scherm mee. Normaal bevinden de sprite pointers zich op de adressen 2040-2047. De sprite pointers bevinden zich nu 17 adressen na het laatste schermadres (2023). Dit is steeds het geval, dus als het scherm verplaatst wordt, bevinden de sprite pointers zich steeds 17 adressen na het laatste schermadres. Als voorbeeld verplaatsen we het scherm naar het laatste gebied in bank 0. Het eerste schermadres is dan adres 15360 en het laatse schermadres is adres 16359.

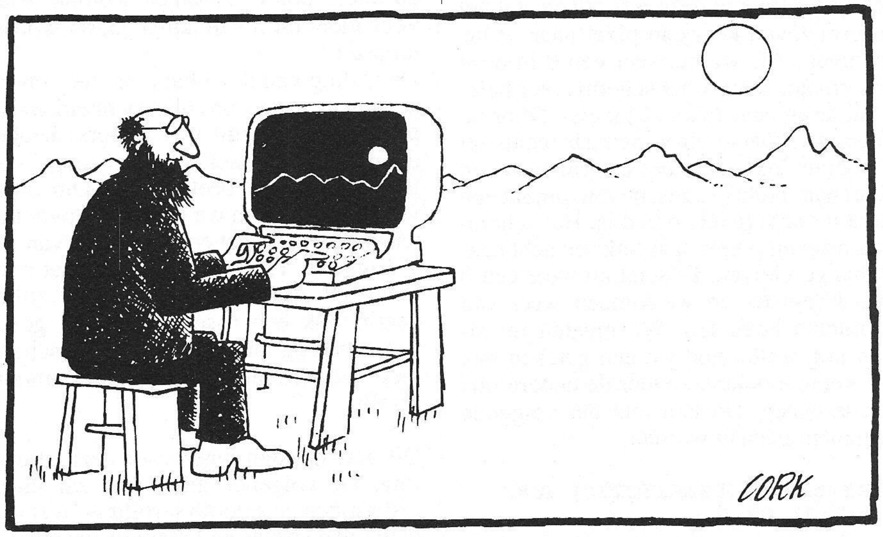
De sprite pointers bevinden zich vanaf adres 16359+17 = 16376.

Nu moet u opppassen als u van bank verandert. Stel, het scherm bevindt zich in het tweede gebied (1024-2023). Nu gaat u van bank 0 naar bank 2. het scherm zal zich dan nog steeds in het tweede gebied bevinden, echter nu in bank 2! De sprite pointers zullen zich dan nu bevinden vanaf adres 32768+2023+17 = 34808.

Samenvattend kunen we zeggen dat u per bank 256 gebieden heeft om een sprite in te zetten. Alleen in de gebieden waar karakterROM is, kunt u geen sprite neerzetten. Het gebied waar u een sprite heeft gezet moet u aan de computer doorgeven door middel van sprite pointers. Deze bevinden zich standaard op de plaatsen 2040- 2047. De sprite pointers verhuizen echter met het scherm mee, als deze wordt verplaatst.

Het high resolution

Toen we het over hires hebben gehad, hebben we gezegd dat hires kan worden aangezet met de volgende twee pokes:




```
POKE 53265, PEEK(53265) OR
    32:
POKE 53272, PEEK(53272) OR 8
```

Met bit 5 in adres 53265 kunt u het hires aan en uit zetten. Als dit bit aan staat, is het hires aan, anders uit. Met het derde bit in adres 53272 kunt u de plaats van het hires in de bank kiezen. Om een hires scherm op te slaan zijn er 25*40*8 = 8192 bytes nodig. In een bank passen dus precies twee hires schermen. We geven weer de situatie voor bank 0. In deze bank loopt het eerste gebied van 0-8191 en het tweede gebied van 8192-16383. De verdeling van de andere drie banks in twee gebieden, gaat equivalent aan de verdeling van bank 0 in twee gebieden. Het vierde bit in adres 53272 is als het ware een soort hires pointer. Als dit bit uit staat, kijkt de VIC naar het eerste gebied in de bank. Staat het bit aan, dan kijkt de VIC naar het tweede gebied.

U moet wel bedenken dat het geen zin heeft om het eerste gebied in bank 0 te kiezen om een hires plaatje in op te slaan. In de onderste vier blokken (0-1023) staat namelijk allerlei informatie voor het besturingssysteem. Als u in dit gebied een hires plaatje zet slaat de computer vast.

U heeft waarschijnlijk wel gemerkt dat adres 53272 veel gebruikt wordt. We geven nog even een overzicht van dit adres:

```
53272:
   bit 0  : niet gebruikt
   bit 1-3: pointer naar
    karakterset
   bit 3  : hires pointer
   bit 4-7: scherm pointer
```

U ziet dat bit 3 van adres 53272 dubbel wordt gebruikt. Hier moet u rekening mee houden als u tegelijk met zelf gedefinieerde karaktersets en hires schermen werkt. Stel u heeft een karakterset gedefinieerd en zet die ergens in het geheugen, bijvoorbeeld op adres 2048. De bits 1-3 van adres 53272 hebben dan het volgende bitpatroon: 001 (zie figuur 2). Bit 3 staat dan uit. Als u vervolgens een hires plaatje, dat zich bevindt op de adressen 8192-16383, aan wilt schakelen, dan moet u bit 3 van adres 53272 aanschakelen. Dit betekent dat als het hires aan staat, u geen beschikking meer heeft over de karakterset die u heeft gemaakt. Dit is ook niet nodig, maar als u het hires weer uitschakelt, moet u wel weer bit 3 van adres 53272 op 0 zetten. Doet u dit niet, dan heeft u geen beschikking meer over uw karakterset en ziet u een hoop troep op het scherm. Dit geldt trouwens ook als u met de standaard karakterset werkt. Deze begint op adres 4096 en dus moet bit 3 van adres 53272 op 0 staan.

Scrolling

In de eerste aflevering van deze cursus hebben we de karakters behandeld. Aan het eind van die aflevering hebben we een klein beetje over animatie van karakters verteld. Het effect van beweging werd verkregen door een karakter steeds uit te wissen en op een aangrenzende positie neer te zetten. Deze vorm van animatie heeft betrekking op een enkel karakter; we beschouwden de karakters als onafhankelijke objecten. We kunnen het scherm met karakters echter ook als een geheel zien. Het scherm kan dan als geheel in horizontale of vertikale richting verschoven worden. Het verschuiven van een heel scherm wordt scrolling genoemd. Er zijn twee soorten scrolling: coarse scrolling en smooth scrolling.

Coarse scrolling

Bij coarse scrolling wordt het scherm verschoven over de breedte of hoogte van een karakter. Dit gebeurt bijvoorbeeld, als u de RETURN-toets drukt terwijl de cursor op de onderste regel staat. De bovenste regel verwijnt dan van het scherm en alle andere regels schuiven een plaats omhoog. Bij coarse scrolling beweegt het scherm schokkerig en is daardoor alleen geschikt voor het scrollen van tekstschermen (bijvoorbeeld bij tekstverwerkers).
Deze vorm van scrolling is vrij eenvoudig te programmeren. Om het scherm bijvoorbeeld naar links te scrollen, moeten op elke regel alle karakters, behalve de eerste, een plaats naar links worden geschoven. De laatste plaats van elke regel komt nu vrij. Op deze plaats kan dan een nieuw karakter gezet worden. Door deze procedure telkens weer te herhalen, kunnen we een tekst over het scherm laten scrollen. Helaas kost het scrollen van een heel scherm nogal veel tijd; er moeten immers 1000 karakters verplaatst worden. Daarom moet scrolling in machinetaal geprogrammeerd worden. Omdat we toch een Basic voorbeeld willen laten zien, beperken we ons nu tot het scrollen van een regel. Er hoeven dan maar 40 karakters verschoven te worden. Het volgende programma scrollt een lichtkrant op de bovenste regel van het scherm.

```
5 rem coarse scrolling
10 for i=1024 to 1062
20 poke i,peek(i+1)
30 next
40 read a
50 if a=-1 then restore:goto
   40
60 poke 1063,a
70 goto10
100 data
    3,15,13,13,15,4,15,18,5,32,
    32,-1
```

In regel 10-30 wordt de bovenste regel een karakter naar links gescrolld. In regel 100 staan de schermcodes van de tekst "commodore " afgesloten door -1. Regel 40-50 leest steeds een schermcode. Deze code wordt dan in regel 60 op adres 1063 gepoked; dit is de positie van het laatste karakter op de bovenste regel. Tenslotte wordt naar regel 10 gesprongen en begint alles weer opnieuw.
In het bovenstaande voorbeeld hebben we aangenomen dat alle karakters dezelfde kleur hebben. Als dit niet het geval is, dan moeten ook de kleuren worden verschoven. Dit gaat op dezelfde manier als het verschuiven van karakters. Voor de bovenste regel moet de FOR-NEXT lus dan lopen van 55296 tot 55334.

Smooth scrolling

Bij smooth scrolling wordt het scherm slechts een pixel per keer verschoven. Met smooth scrolling kan een mooie soepele beweging van het scherm worden gemaakt, die zeer geschikt is voor videogames. Er kan bijvoorbeeld een berglandschap over het scherm worden gescrolld, terwijl een sprite in de vorm van een vliegtuigje in het midden van het scherm blijft staan en alleen op en neer kan bewegen. Het lijkt dan net of het vliegtuigje over het landschap heen vliegt.
Om op het scherm een pixel te verschuiven, zijn er twee registers beschikbaar. Een voor een horizontale en een voor een vertikale verschuiving. Het register voor de vertikale verschuiving bevindt zich op adres 53265. Het register voor horizontale verschuiving heeft adres 53270. In beide registers zijn alleen bits 0-2 voor scrolling. Met deze drie bits kunnen de getallen 0 tot 7 gerepresenteerd worden. We zullen nu laten zien hoe het scherm met behulp van smooth scrolling naar rechts geschoven kan worden.
Standaard staat het getal 0 in bits 0-2 van adres 53270. Als we achtereenvolgens de getallen 1 tot 7 in deze bits poken, zal het scherm zeven keer een pixel naar rechts schuiven. Als we nu weer een 0 in deze bits poken, schuift het scherm weer helemaal terug naar links (7 pixels). Door nu direct een coarse scrolling naar rechts uit te voeren, lijkt het of het scherm weer een pixel naar rechts is geschoven, omdat een karakter acht pixels breed is. Het scherm is dan zeven pixels naar links en acht naar rechts geschoven. Er staat nu weer een 0 in het register en we kunnen weer van voorafaan beginnen. We moeten nu alleen nog weten hoe we een getal in bits 0-2 kunnen poken, zonder de andere bits aan te tasten. Dit kan met het volgende statement gedaan worden:

```
POKE 53270,(PEEK(53270) AND
    248) OR G
```

Hierbij is G het getal tussen 0 en 7. Om het scherm naar links te scrollen, moet er eerst een 7 in het register worden gezet en vervolgens de getallen 6 tot 0. Daarna moet er weer een 7 in gepoked worden en moet een coarse scrolling naar links uitgevoerd worden. Het register voor vertikale scrolling werkt analoog. Als van 0 tot 7 wordt geteld, scrollt het scherm omlaag.
De coarse scrolling moet nu heel snel gebeuren. Als we het hele scherm echter vol zetten met hetzelfde karakter, hoeft er helemaal geen coarse scrolling gedaan worden. Het scherm ziet er na coarse scrolling toch het zelfde uit als voor de scrolling. We kunnen dan een Basic programmaatje maken dat smooth scrolling laat zien.

```
5 rem smooth scrolling
10 for i=1024 to 2023
20 poke i,204:next
40 for i=0 to 7
50 poke 53270,(peek(53270)
   and 248) or i
60 next:goto40
```

Regel 10-20 zet het scherm vol met het zelfde karakter. Regel 40-60 zorgen voor de scrolling. Nu blijkt echter dat de zijkant van het scherm steeds heen en weer trilt. Dit kan verholpen worden door de zijborders een stukje breder te maken, waardoor het trillende gedeelte onder de border verdwijnt. Door bit 3 van adres 53270 uit te zetten worden de zijborders breder. Er zijn dan nog maar 38 kolommen op het scherm in plaats van 40. Als u in het bovenstaande programma de volgende regel toevoegt, is het trillen verholpen.

```
30 poke 53270,peek(53270) and
   255-2~3
```

Bij vertikale scrolling moeten de boven en onder border verbreed worden. Hiervoor moet bit 3 van adres 53265 worden uitgezet.
De trilling van de zijkant van het scherm is niet de enige onvolkomenheid in het programma. U zult wel gemerkt hebben dat de scrolling niet helemaal soepel verloopt, maar een beetje golft. Om dit te verhelpen moeten we iets weten over rasterlijnen en de scherm opbouw van het TV scherm. Hierover zullen we het in de volgende aflevering hebben. We zullen daarin ook een assembly listing geven van een programma dat een heel scherm met verschillende karakters smooth scrollt.

Dit was het dan weer voor deze aflevering. De volgende aflevering zal -naast rasterlijnen en smooth scrolling- over animatie van sprites en karakters gaan.

Basic miniatuurtjes

Een rubriek van Alex van Maarschalkerwaart

Hires-Tekenen.

Met een viertal grote programmaregels heeft Rob Hagelstein uit Doetinchem wel een mooie oplossing gevonden om X-Y-coördinaten van formules om te zetten en te rangschikken in Bitpatronen.

```
10  s=53265:forc=1024to2023:pokec,22:n
    ext:dimk(2800):pokes+7,25:pokes,18
    7
20  t=t+.002:x=160+55*(cos(t)+cos(25*t
    )):y=100+45*(sin(t)+sin(25*t))
30  p=8192+(xand504)+(yand7)+(yand248)
    *40:pokep,peek(p)or2[VERTIKALE PIJ
    L](notxand7)
40  onpeek(198)+1goto20:printchr$(147)
    :pokes+7,21:pokes,27
```

Door gebruik te maken van AND,OR en NOT opdrachten is het resultaat dat EEN programmaregel vele normaal gebruikte methoden vervangt. In regel 30 staat de belangrijke formule die de X-Y-coördinaten omrekent naar het adres van een bepaalde bitpatroon in het hires-scherm. De POKE zorgt ervoor dat de juiste bit in het gerekende adres zal oplichten. Regel 20 is de wiskundige formule-regel van de figuur dat op het scherm getekend moet worden.
Wil je dus een zelfbedacht figuur op het scherm teken dan moet deze formule dus vervangen worden. Enkele voorbeelden van tekeningen.

Voorbeeld 1:Rechte schuine lijn.

```
20  if x<319thenx=x+1:y=x
```

Voorbeeld 2:Sinus functie.

```
20  if x<319then x=x+1:y=100+80*sin(x/25)
```

Voorbeeld 3: Cirkel

```
20  t=t+0.02:x=160+90*sin(t):y=100+80*cos(t)
```

Voorbeeld 4: Rechthoek

```
20  xe=xe-(ye=0)+(ye=100):ye=ye-(xe=100)+
    (xe=0):x=xe+110:y=ye+50
```

Voorbeeld 5: Dubbele sinus functie

```
20  if x<319thenx=x+0.5:y=100+80*sin(x
    /25)+15*sin(x/5)
```

Voorbeeld 6: Verdraaide cirkel

```
20  t=t+0.03:x=160+(120-t*2)*sin(t):y=
    100+80*cos(t)
```

Voorbeeld 7: Gecirkelde sinus (zie foto)

```
20  t=t+0.1:x=12+t*2+20*sin(t):y=100+3
    5*sin(t/10)+20*cos(t)
```

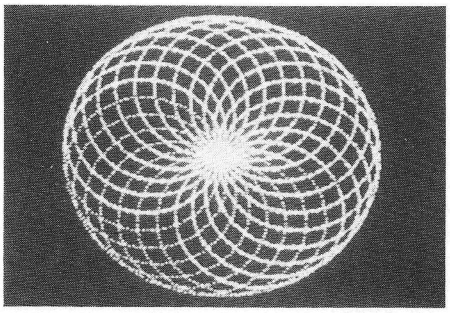
Voorbeeld 8: Teken-prG. met joystick.
Een eenvoudig te gebruiken programma wat alleen als klein nadeel heeft dat het tekenen traag gaat.

```
5   x=156:y=100:p=8192
10  s=53265:forc=1024to2023:pokec,22:n
    ext:dimk(2800):pokes+7,25:pokes,18
    7
20  j=peek(56320):x=x-(jand8)/8+(jand4
    )/4:y=y-(jand2)/2+(jand1)
25  ifj>111thenpokep,0
30  p=8192+(xand504)+(yand7)+(yand248)
    *40:pokep,peek(p)or2[VERTIKALE PIJL]
    (notxand7)
40  onpeek(198)+1goto20:printchr$(147)
    :pokes+7,21:pokes,27
```

Lazer effect.

Voor de C 128 gebruikers die van schieteffecten houden, heeft Peter Verdock uit St-Pieters-leeuw in België een mooi en kort programma gemaakt dat het proberen waard is.

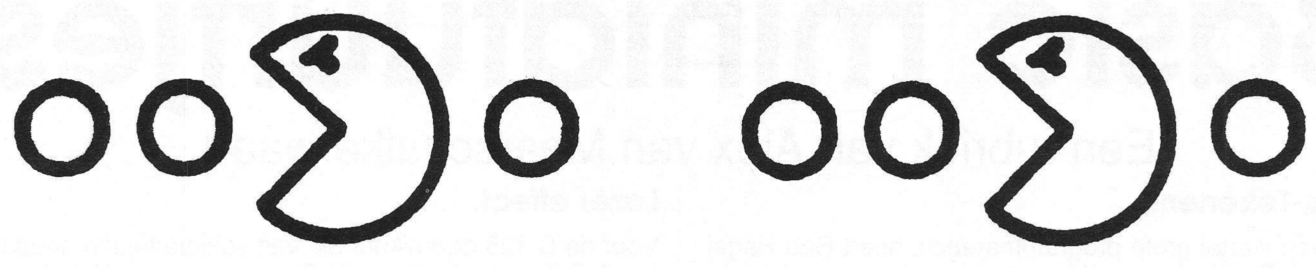
```
0   rem voor c 128
1   rem ****lazer effekt****
2   rem door verdock peter
3   rem zonneweelde 57
4   rem 1600st-pieters-leeuw
5   rem       belgie
6   rem tel(0032)2 377.58.27
10  color 0,2:color 1,4:color 4,14
20  graphic 1,1
30  for a=10to 150step 10
40  for b=150to 10step -10
50   1,a,b to 80,160
60  next b
70  next a
80  char 1,20,13,"lazer[SPACE]effekt
90  sleep 10
100 graphic 0,1
```



Linkerbovenhoek.

Dit is een programma dat kijkt naar wat er in de linkerbovenhoek staat.
Als daar een [VERTIKALE PIJL] staat,verandert de cursor van kleur, als er een * staat, verandert de border van kleur. Het programma staat in de interrupt en is gemaakt door Peter Buijsman uit Beverwijk.

```
5   rem--linkerbovenhoek--
10  restore
20  fort=0to43:reada:poke49152+t,a:o=o
    +a:nextt
30  ifo<>4530thenprint "fout[SPACE]in[
    SPACE]data":end
40  data120,169,13,141,20,3,169,192,14
    1,21,3,88,96,162,30,236,0,4,240,3,
    76
50  data29,192,238,134,2,76,49,234,162
    ,42,236,0,4,240,3,76,49,234,238,32
60  data208,76,49,234
70  rem
80  rem
90  rem * in linkerboven hoek is rand
100 rem [VERTIKALE PIJL] in linkerboven
    hoek is cursor
110 rem (her)start met sys 49152
120 rem
130 rem gemaakt door peter buijsman
140 sys49512
```



Wordsort.

Dit programma rangschikt een te geven aantal woorden alfabetisch.

```
5   rem --wordsort--
10  rem --sven verstrepen--
20  printchr$(147)
30  input"hoeveel[SPACE]woorden";r:dim
    a$,r:print
40  forx=1tor:print"woord";x:inputa$(x
    ):next
50  forx=r-1to1step-1
60  fory=1tox
70  ifa$(y)>a$(y+1)thenz$=a$(y+1):a$(y
    +1)=a$(y):a$(y)=z$
80  nexty
90  nextx:print
100 forx=1tor
110 printa$(x)
120 nextx
130 print:end
```

Secondenomzetter.

Dit programma zet een op te geven aantal seconden om in uren, minuten en seconden.

```
5   rem --secondenomzetter--
10  rem--sven verstrepen--
20  printchr$(147)
30  input"aantal[SPACE]seconden";t
40  u=int(t/3600)
50  rs=t-(u*3600)
60  m=int(rs/60)
70  s=rs-(m*60)
80  printu"uur"
90  printm"minuten"
100 prints"seconden"
```

Grafiek.

Dit programma tekent een grafiek van de getallen in de dataregel-260.

```
40  rem --grafiek--
50  rem --sven verstrepen--
100 fory=80to0step-5
110 printtab(5-len(str$(y)));y;
120 forx=1to5
130 readp
140 ifp>=ythenprint"[SPACE][CTRL-9][SP
    ACE][CTRL-0][2xSPACE]";:goto160
150 print"[4xSPACE]";
160 nextx
170 print
180 restore
190 nexty
200 rem*jaargetallen afdrukken*
210 printtab(6);
220 forj=60to80step5
230 printj;
240 nextj
250 print
260 data0,10,25,30,30
270 end
```

Berglandschap SB.

Dit is een Simons-Basic programma dat steeds een ander berglandschap tekent.

```
1   rem--berglandschap--
2   rem--sven verstrepen--
3   rem--in simos's basic--
10  colour6,6
20  hires5,6
30  y=100
40  forx=0to320
50  r=int(rnd(1)*3)+1
60  ifr=2theny=y:plotx,y,1:plotx,y+1,1:plotx
    ,y-1,1
70  ifr=1theny=y-1:plotx,y,1:plotx,y+1,1:plotx
    ,y-1,1
80  ifr=3theny=y+1:plotx,y,1:plotx,y+1,1:plot
    x,y-1,1
90  nextx
100 paint0,160,1
110 pause3
```

Reactiesnelheid.

Dit is een reactie spel dat niet van valsspelen houd.

```
1   rem--reactiesnelheid--
2   rem--sven verstrepen--
10  print"[SHIFT-CLR]"
20  print"druk[SPACE]toets[SPACE]in[SP
    ACE]na[SPACE]de[SPACE]flits"
30  print"houd[SPACE]u[SPACE]klaar..."
40  forx=1to700
45  getv$:ifv$<>""then print"niet[SPAC
    E]valsspelen":ford=1to200:next:got
    o10
46  nextx
50  poke53280,1:poke53281,1
51  poke53280,7:poke53281,7
55  ct=ct+1
60  geta$:ifa$=""then55
70  print"uw[SPACE]reaktiesnelheid[SPA
    CE]is";ct
```

Kleurstick.

Met dit miniatuurtje kun je wanneer je maar wilt de kleuren van het scherm veranderen.
Het instellen gaat met de joystick in poort 2.
Op deze manier hoeft u dus geen lange commando's meer te geven, aldus Ernst-Jan Bostelaar uit Voorburg.

```
1   rem---kleurstick---
2   rem--ernst-jan bostelaar--
10  fort=0to28:reada:c=c+a
20  poke49152+t,a:next
30  ifc<>3246thenprint"data-fout!":end
40  sys49168
50  data173,0,220,201,127,240,6,141,32
60  data208,141,33,208,76,49,234,120,1
    69
70  data0,162,192,141,20,3,142,21,3,88
    ,96
```

Voel je een Sim in Simcity

of ga met Kuifje naar de maan

Infogrames heeft ons in het verleden reeds verbaasd met originele spellen van topkwaliteit. Wederom moet ik constateren dat zij zichzelf overtroffen heeft. Dit keer gaat het niet om een aktiespel, maar om een simulator. Doel van de simulator is het ontwerpen, beheren en onderhouden van je eigen droomstad. Dit spel is op normaal geldende regels gebaseerd. Hierbij spelen de volgende factoren een belangrijke rol:

- de menselijke factor wordt beïnvloed door woonruimte en vacatures;
- de economische factor wordt bewerkt door de aanwezige industriële en commerciële ruimte, werkloosheid, energie, belastingen en handelsopties;
- de overlevingsfactoren zijn afhankelijk van de mogelijkheden om rampen, misdaad en verontreiniging in te dammen; de politieke factoren zijn afhankelijk van de publieke opinie en het tevreden stellen van de bevolking en de commerciële sectoren.

Dit doel kan binnen de simulator bereikt worden door de stad te bouwen met de beschikbare gereedschappen (iconen). Het spel is met verschillende gereedschappen uitgerust, zoals de bulldozer die het land klaarmaakt voor de bouw. De residentie-, commerciële en industriële iconen wijzen de plaatsen aan waar zij gebouwd kunnen worden. Tevens zijn er iconen voor de bouw van wegen, treinen, parken, politie- en brandweerbrigades, energiecentrales, havens, vliegvelden en een sportveld. Door ervoor te zorgen dat de energie een bouwzone bereikt, zal er worden gebouwd en leven ontstaan. Let wel, de speler wijst alleen aan!

Het is de Sim (inwoner van Simcity) die bouwt en het gebouw renoveert naarmate er meer inwoners zijn. Verder vormt de Sim een gewoon huis om in een flat. Een 'vraag'-window geeft aan naar welke gebouwen veel vraag is.

Conclusie: Ik beschouw dit programma niet alleen als een zeer boeiende simulator, maar ook als een eerste klas educatief programma voor een studie in wat men duur noemt: Planologie.

Het programma is goed afgewerkt en een nuttige uitbreiding op zowel scholen als huiskamers. Men hoeft zich geen zorgen te maken wat betreft het systeem, want behalve voor de Amiga is het tevens verkrijgbaar voor de PC (EGA en VGA), Atari ST, Mac 2 en de Commodore 64.

Beoordeling

Spel	95
Beeld	90
Afwerking	90
Totaal	92

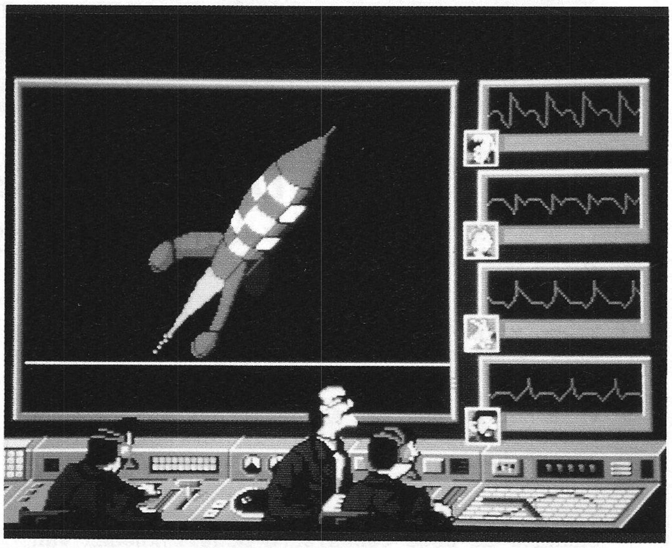
Kuifje, raket naar de maan

Dit spel is het tweede spel waarbij Infogrames gebruik maakt van een populaire stripserie. Naar alle waarschijnlijkheid zal hij populairder worden dan zijn voorganger North en South, daar hij het meer van actie moet hebben.

Het verhaal: in Syldavië staat de eerste raket naar de maan klaar voor lancering. De raket, ontworpen door professor Zonnebloem, zal Kuifje en zijn trouwe viervoeter met Kapitein Haddock en Ingenieur Wolff naar de maan brengen. De raket zal in vijf fases de maan bereiken. Iedere fase bestaat uit twee delen. In het eerste deel moet men meteoren ontwijken en gekleurde ballen vangen. Van de ballen zijn er twee kleuren. De gele bevatten de noodzakelijke energie om te kunnen vliegen en de rode zijn nodig om vooruit te komen. Per level moet men 8 rode ballen vangen.

Het tweede deel van het spel speelt zich af in de raket. Hier moeten 4 doelen bereikt worden. Men moet de branden blussen die door ene Kolonel Boris gemaakt worden, men moet de geplaatste explosieven ontmantelen, men moet de medepassagiers bevrijden en kolonel Boris moet gevangen worden. Dit alles voordat de energie naar 0 gedaald is en de explosieven tot ontsteking komen.

Op de onderste regel ziet men hoeveel bommen er nog onschadelijk gemaakt moeten worden. Alle acties worden ondernomen door de persoon of het object aan te raken met uitzondering van Kolonel Boris: deze moet je van achter grijpen. Naarmate men een hoger level bereikt wordt ook het speelveld in de raket groter en dus moeilijker. Als je ergens naar toe wilt zweven kun je met <F1> de zwaartekracht uitschakelen, hetgeen een komisch manoeuvreerprobleem geeft.



Het spel is mooi van beeld en geeft de tekenstijl van Hergé goed weer. Het spel is verkrijgbaar voor de Commodore 64 en Amiga, evenals voor de Atari ST, Amstrad, Spectrum en PC.

Beoordeling

Beeld	80
Spel	80
Afwerking	85
Totaal	82

Lawrence van Rijn

RAM 1764

een verloren zoon keert terug

De RAM uitbreidingen 1764 en 1750 zijn sinds 1986 van de markt verdwenen. Of dit nu te maken had met het steeds maar duurder worden van de RAM chips of dat er geen afzetgebied genoeg was voor dit artikel, wie weet.

Wat we wel weten is dat de 1764 REU (Ram Expansion Unit) weer volop verkrijgbaar is. Een verloren zoon keert terug.

Meer Snelheid

Heb u weleens in de Photo Manager een kopie gemaakt van een nogal grote clipart? Dat duurt eeuwen. Alsof de tekening pixel voor pixel van disk wordt gelezen. Er is natuurlijk iets aan te doen. Een REU (RAM Expansion Unit) zou zo'n oplossing zijn. Met een REU type 1764 achterin uw C64 of C128 krijgt u het gevoel als dat van een Eend-rijder die een V8 motor onder zijn motorkap heeft zitten. Dat speciale TURBO gevoel. Met een REU geen tijd meer voor het drinken van een kop koffie terwijl GEOS vanuit een applikatie terugkeert naar de deskTop. Snelheid was een luxe die niet voor de Commodore gebruikers was weggelegd. Tot nu....
De RAM uitbreiding is volop verkrijgbaar op dit moment en het gebruik van deze REU 1764 heeft tot een aantal interessante ontdekkingen geleid.

1750 Moederbord

Allereerst voor de techneuten onder u: de Reu 1764 die momenteel op de markt is heeft het moederbord van een REU 1750, dit is het model dat oorspronkelijk voor de C128 modellen werd vervaardigd. In deze 1750 zitten 16 RAM chips nr. 41256, een besturingsRom en het bord levert 512 KB extra RAM geheugen op voor de C128 gebruiker.
Plaats je nu de helft van het aantal chips, dan levert dat 256 KB extra geheugen op. Dit is nu precies gedaan met de huidig leverbare RAM uitbreiding 1764.

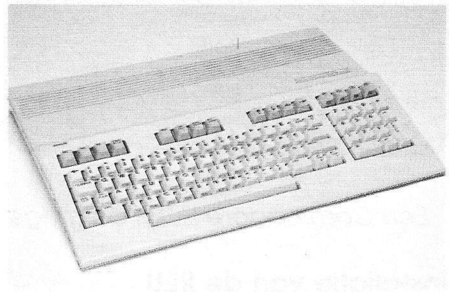


Voor grafisch werk is voldoende RAM een must

8 RAM Chips in plaats van 16. Nu hoor ik de soldeergrage techneut al denken, "Simpel, 8 chips bijzetten, jumpertje doorsnijden en ik heb 512 KB". U heeft gelijk ook. Dat is alles wat u hoeft te doen. Maar hier is natuurlijk enige voorzichtigheid geboden.
Voor diegenen die nooit een soldeerbout vasthouden is dit niet zo simpel. Wel is er een bonus op de koop toe, want de 512 KB heeft u niet alleen op de C128 extra maar ook op de C64. En dat is nieuw. Want werd ons niet altijd verteld dat de 1764 voor de C64 was en de 1750 voor de C128?
Denkt u eens na, 512 KB extra geheugen in een C64. Een PC heeft ook minimaal 512 KB werkgeheugen.... Werkgeheugen? Extra geheugen? Dat is niet hetzelfde. Of wel?

RAMdisk

Neen, er is wel degelijk verschil tussen werkgeheugen en een extra geheugen. Een werkgeheugen is de RAM die direkt adresseerbaar is door de CPU. Een extra geheugen is daarentegen toegevoegde RAM die door bepaalde waarden in de stack adresseerbaar wordt.
Een RAMdisk is daar een goed voorbeeld van.



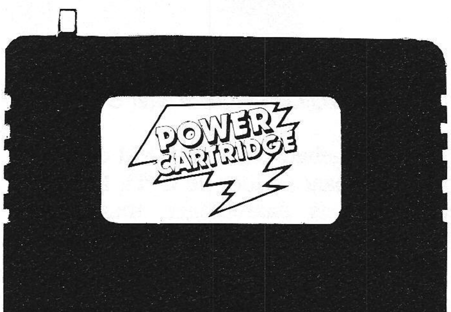
Met een RAMdisk wordt een bepaald gedeelte van het geheugen gereserveerd voor tijdelijke opslag van bijvoorbeeld een programma dat u vaak gebruikt.

In plaats van dat dit programma elke keer van disk wordt gelezen, wordt dit nu van-

uit deze gereserveerde RAM-disk (gereserveerde geheugenruimte) ingelezen. Hierbij is tijd de grootste winst.
Het programma is al aanwezig voordat u een keer met de ogen heeft geknipperd.

RAM en ROM

De RAM-uitbreiding 1764 is ook extra RAM, maar dan in de vorm van een cartridge. Nu kennen we het fenomeen cartridge al enige tijd. De eerste cartridges werden door Commodore zelf uitgebracht. Dit waren de ROM-cartridges.
Het programma zat kant-en-klaar ingebakken en was meteen aanwezig als de computer werd aangezet. Zoals de naam al zegt, zaten deze programma's in ROM. ROM betekent Read Only Memory, wat wordt omschreven als 'alleen lezen en niet schrijven'.
In een later stadium verschenen de Epromcartridges en Eprom-banken. Men kon nu zelf bepalen welke programma's er in deze cartridges kwamen. De Eprombanken werden groter en groter. Het begon met 8KB naar 16KB en via 32 en 64KB, naar de welbekende 256KB Epromkaart. Er is zelfs een 1 Megabyte Epromkaart, beter bekend als de Goliathkaart.
Maar het blijven ROM-cartridges en u kunt deze alleen uitlezen.
Een extra RAM-geheugen daarentegen kunt u zowel uitlezen als beschrijven. De REU 1764 is dan ook een RAM-cartridge.



Een Commodore ROM cartridge

Installatie van de REU

Als u de REU 1764 uitpakt zult u zien, dat dit een keurige nette kast is die u kunt aanbrengen in de Expansion poort van uw C64 of C128. Wat we bij de meeste Epromkaarten misten, zit hier keurig op z'n plaats: een ondersteuningsvoetje. Het is een keurig afgewerkt geheel. Op de bijgeleverde diskette staat op de A-kant het installatieprogramma. Het kan gestart worden met load "*",8,1. U ziet een menu met verschillende opties, waarvan het eerste het installatieprogramma is, dat de RAM 1764 als drive 9 installeert. U heeft nu een extra 1541 diskdrive ter beschikking en kunt dan ook de software die 2 drives ondersteunt dan ook als zodanig gebruiken.
Vergeet niet dat het uiteindelijk een tijdelijk geheugen is en dat na uitschakeling van de stroomtoevoer alles verloren gaat. Er moet dan ook altijd een back-up gemaakt worden als u stopt. Dat brengt ons bij de volgende menu optie, een file kopiëerprogramma. Deze optie spreekt natuurlijk voor zichzelf. Er is vervolgens een optie om het RAM te testen en er zijn twee demofiles die gemak en snelheid van de RAM-uitbreiding tonen.
Zoals u waarschijnlijk al heeft begrepen, zijn dit Basic programma's en dat heeft niets met GEOS te maken. Op de achterkant van de diskette staat het Upgrade programma van GEOS 1.2 naar GEOS 1.3. U kunt dit programma wel vergeten, want een beetje GEOS-gebruiker werkt al met GEOS 2.0 (en zeker met versie 1.3) en zal deze upgrade daarom niet nodig hebben.

Installeren met Configure

Het installeren van de RAM 1764 onder GEOS wordt gedaan met de Configure-file die op de SYSTEM disk staat. U gaat als volgt te werk:
Start GEOS op zoals u dat gewend bent. Als de deskTop eenmaal op het scherm staat, ziet u dat de RAM uitbreiding nog niet op het scherm als Drive B of C staat. U zult GEOS nog moeten vertellen, dat er een RAM-uitbreiding bij is gekomen. Dit gebeurt met de Configure-file. U klikt het Configure-file icon tweemaal aan en deze zal dan op het scherm verschijnen. U ziet een scherm, verdeeld in 4 gelijke vakken :Drive A, Drive B, Drive C en RAM-Expansion geheten. In het vak RAM-Expansion ziet u dat er 256 KB aanwezig is en dat de vakjes achter DMA for 'MoveData' en RAM Reboot al geselecteerd zijn.
In het vak van Drive B staat het hele assortiment te selecteren drives. Hier kiest u voor RAM 1541. Nu kiest u in het commandomenu FILE voor Save Configuration en de door u gekozen configuratie wordt als zodanig op de disk geschreven. Nogmaals in het FILE menu kiest u nu voor QUIT en keert u terug naar de deskTop. Dit installatieproces hoeft u maar één keer te doen. De volgende keer dat u GEOS opstart zal uw configuratie automatisch worden ingelezen en verschijnt de RAM-uitbreiding als drive B op de deskTop. Terugkomend op de twee opties in het RAM-Expansion vak. DMA for MoveData stelt GEOS in staat grote hoeveelheden data te transporteren van disk naar het werkgeheugen. Dit wordt mogelijk gemaakt, doordat een virtuele 1541-drive wordt geselecteerd. Dit houdt in, dat er op een RAM 1541 een beschikbaar geheugen van 165 KB aanwezig is; net zoveel als een echte 1541 diskdrive.
Het resterende geheugen, tot 256 KBm, wordt door GEOS gebruikt voor enkele permanente routines in op te slaan. Zo worden de Configure preferences permanent in het RAM opgeslagen en wordt er een buffer gereserveerd voor snelle dataoverdracht naar disk. Tevens is er ruimte gereserveerd voor het opslaan van de directories van de aanwezige diskdrives.

Tot slot

GEOS 2.0-gebruikers hebben de mogelijkheid om voor een 'Dir Shadowed' diskdrive te kiezen. Bij deze keuze wordt bij de eerste keer dat de directory van een diskdrive wordt gelezen, deze opgeslagen in het RAM. Bij de volgende lees opdracht wordt de directory gelezen vanuit RAM en niet vanaf disk. Wordt er naar de diskdrive geschreven, dan wordt de directory aangepast.
De RAM Reboot-keuze in het RAM-Expansion vak stelt de gebruiker in staat na een reset met de RESTORE toets alsnog het geheugen van het RAM veilig te stellen. Is dit niet mogelijk, dan kan men met het programma Rboot, dat op de SYSTEM disk staat de deskTop opnieuw inlezen en dan alsnog het geheugen in het RAM veilig stellen.

De RAM-uitbreiding wordt door GEOS op allerlei manieren ondersteund. Voor GEOS-gebruikers is het eigenlijk een must.
Voor meer informatie en verkrijgbaarheid over deze RAM-uitbreiding kunt u terecht bij de Stichting GEOS Gebruikers.

Pim Broekhuijzen.

Tips & Trucs 64

Een gure wind waait om het huis, hagelstenen slaan tegen de ruiten en een dik pak sneeuw weerhoudt u ervan om naar buiten te gaan. Kortom, prima weer om met uw 64 voor de kachel te gaan zitten experimenteren met Tips & Trucs 64. Deze keer bieden wij de volgende onderwerpen: besturing van de cursor, border killer, uitlezen van het disk error kanaal en negatieve getallen in het tweecomplement systeem.

De cursor

In deze aflevering van Tips & Trucs zullen we wat extra aandacht besteden aan de cursor. Dit knipperende blokje op het scherm geeft de plaats aan, waar het volgende karakter dat ingetypt wordt, op het scherm komt te staan. Het knipperen van de cursor wordt bereikt door het karakter dat 'onder' de cursor staat, respectievelijk eerst gewoon op het beeld te zetten, en daarna in reverse video op het beeld te zetten. Door dit steeds om en om te doen, wordt het effect van een knipperende cursor bereikt.
Het afbeelden van een karakter in reverse video wordt gedaan door bit 7 aan te zetten van de geheugenplaats waar het karakter staat. We geven een voorbeeld. Op het scherm linksboven staat een willekeurig karakter, bijvoorbeeld een spatie. Het karakter linksboven op het scherm correspondeert met geheugenadres 1024. Door nu bit 7 van adres 1024 aan te zetten komt het karakter linksboven in reverse video te staan. Het aanzetten van bit 7 van adres 1024 kan gedaan worden door: POKE 1024, PEEK(1024) OR 128.
Als een spatie (schermcode 32) wordt geïnverteerd, in reverse video wordt gezet dus, krijgen we het karakteristieke blokje van de cursor. Dit blokje heeft schermcode 32 + 128 = 160. Als u het blokje van de cursor dus wilt herdefiniëren, moet u het karakter met schermcode 160 een nieuw jasje geven.

De cursor aanzetten

Tot nu toe hebben we het gewoon gehad over de cursor in de direkte mode. Vanuit programma's kan echter ook de cursor op het beeld gezet worden. Dit kan gedaan worden met geheugenadres 204. Door in dit adres een 0 te zetten, wordt de cursor aangezet. Met POKE 204, 1 wordt de cursor weer uitgezet.
Een veel gehoord nadeel van de GET instructie is dat de cursor niet op het scherm wordt gezet. Met de vorige POKE kunnen we dit nadeel heel simpel opvangen. Voordat we met de GET instructie invoer van het toetsenbord vragen, zetten we gewoon eerst de cursor op het beeld. Daarna zetten we de cursor weer uit. Hier volgt een voorbeeld van een dergelijk programma:

```
10 rem cursor op het beeld
20 print"j/n";:poke 204,0
30 get a$
40 if a$="" or (a$"j" and
    a$"n") then 30
50 if peek(207)=0 then 50
60 poke 204,1
70 print a$
```

In regel 20 wordt de cursor op het beeld gezet. In regel 30 en 40 wordt gewacht op goede invoer, namelijk een j of een n. Als de invoer goed is, wordt in regel 60 de cursor weer uitgezet en in regel 70 het karakter op het beeld gezet.
We hebben regel 50 nu nog niet behandeld. In adres 207 wordt bijgehouden of de cursor 'uitstaat' of 'aanstaat', oftewel de blinkfase van de cursor. Als de cursor niet blinkt bevat adres 207 een 1, als de cursor blinkt een 0. In regel 50 wordt dus eerst gewacht totdat de cursor niet meer blinkt, alvorens de cursor uit te zetten. Dit voorkomt dat er een karakter in reverse video op het scherm blijft staan.
We kunnen het bovenstaande programma uitbreiden, zodat er een hele string kan worden ingetypt. Op deze manier kunnen we het nadeel van de INPUT instruktie opvangen, namelijk dat er geen komma's ingevoerd kunnen worden. Met het volgende programma kan dit wel. De ingetypte string komt te staan in variabele T$.

```
10 rem verbeterde input
20 poke 204,0
30 get k$: if k$="" then 30
40 if k$=chr$(13) then 110
50 if k$=chr$(20) then 80
60 t$=t$+k$
70 print k$;:goto 30
```



```
80 if t$="" then 30
90 t$=left$(t$, len(t$)-1)
100 print k$;:goto 30
110 poke 204,1
120 sys 59932
130 print
```

In regel 40 wordt gekeken of er een return is gegeven. Als dit zo is, wordt de cursor weer uitgezet, de input van het toetsenbord is dan immers ingetypt. In regel 50 wordt gekeken of de INST/DEL toets ingetypt is. Als dit het geval is, wordt in de regels 80-100 het laatst ingetypte karakter verwijderd uit T$. Sys 59932 op regel 120 test of de cursor nog aan staat. Als dit het geval is, dan wordt de cursor uitgezet.

Cursor bug

Er zitten in de Commodore 64 een aantal vreemde bugs (fouten). Een van die bugs is dat de cursor in bepaalde gevallen buiten het scherm treedt. Als we de cursor naar beneden bewegen met de cursortoetsen, scrollt het beeld naar boven als de cursor helemaal onderaan is. We geven nu een eenvoudig programma, waarbij het scherm niet altijd naar boven scrollt. De cursor gaat dan naar beneden, en komt in het RAM geheugen terecht. Hier komt het programma:

```
10 POKE 204,0
```

```
20 PRINT PEEK(207);:GOTO 20
```

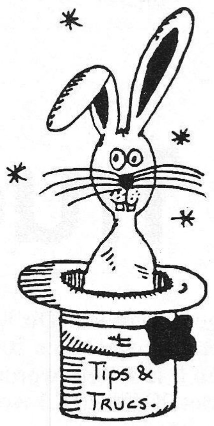
In regel 10 wordt de cursor aangezet. Let wel, de spatie tussen POKE en 204 is essentieel in dit programma! In regel 20 wordt telkens weer de waarde van geheugenadres 207 op het scherm gezet. Zoals al eerder gezegd, wordt er een 1 op het scherm gezet als de cursor blinkt, en een 0 als de cursor niet blinkt. Als dit programma gerund wordt, stopt het na een tijdje met een Undef'd Statement Error in regel 20. Misschien krijgt u deze foutmelding niet meteen als u het programma uitvoert. Of u wacht dan een tijdje, of u drukt op run/stop en start het programma weer op. Na een tijdje zal de foutmelding verschijnen. Als we het programma dan weer bekijken met LIST zien we het volgende:

```
10 POKE 204,0
32788 PRINT PEEK(207);:GOTO
    20
```

Regel 20 is veranderd in regel 32788. Dit komt omdat de cursor buiten het scherm is getreden en een byte heeft veranderd in het geheugen. Het karakter linksonder op het scherm correspondeert met adres 1984. Tellen we bij dit adres 80 (2 regels) op, dan krijgen we adres 2064. Dit is het adres waar de cursor in het geheugen iets heeft veranderd. Het hoogste bit van adres 2064 heeft de cursor aangezet. Immers, het knipperen van de cursor wordt bereikt door bit 7 van het karakter waar de cursor 'op' staat, om de beurt aan en uit te zetten. Nu is bij het bovenstaande programma adres 2064 ook net het hi-byte van het regelnummmer van de tweede regel code. Het lo-byte van het regelnummer van de tweede regel is adres 2063. Daar stond al 20 in. Het hi-byte stond op 0. Nu het hoogste bit in dit byte is aangezet door de cursor, bevindt zich de waarde 128 in dit byte. Samen geven de lo- en hi-byte de waarde 20 + 256*128 = 32788. Als de computer dan de instructie GOTO 20 wilt uitvoeren, kan de computer regel 20 niet meer vinden. Deze regel is immers veranderd in regel 32788, en dit resulteert in de foutmelding Undef'd Statement Error.

Cursor op scherm

Er zijn een aantal adressen in de computer die informatie over de cursor bevatten. We hebben bijvoorbeeld de adressen 204 en 207 al behandeld. Er zijn nog een paar adressen die informatie geven over de cursor.
De adressen 209, 210 en 211 definiëren samen de plaats van de cursor op het scherm. We kunnen met behulp van deze adressen de cursor op een willekeurige plaats op het scherm zetten. Als we dit



willen doen, moeten we als eerste de regel bepalen, waarop de cursor moet komem te staan. Daarbij heeft de eerste regel nummer 0. De laatste regel is regel 24. De waarde van de regel waarop de cursor moet staan, moet worden vermenigvuldigd met 40. Vervolgens moet bij dit getal nog 1024 worden opgeteld. De verkregen waarde moet worden opgesplitst in een lo- en een hi-byte. Dit lo- en hi-byte moeten respectievelijk in de adressen 209 en 210 worden gepoked. Daarna moeten we de positie van de cursor op de regel bepalen. De eerste postie heeft waarde 0 en de laatste waarde 39. Dit getal moet in adres 211 worden gepoked. We zullen nu de algemene formule geven voor het zetten van de cursor op een willekeurige plaats op het scherm. Hierbij geeft Y de regel en X de positie op deze regel.

```
POKE 210, (1024 + 40*Y)/256
POKE 209, (1024 + 40*Y) -
    256*PEEK(210)
POKE 211, X
```

Na het uitvoeren van bovenstaande drie regels, kan er bijvoorbeeld iets met PRINT op de bepaalde positie neer worden gezet.
Als u in assembly programmeert, kunt u het bovenstaande effect veel sneller verkrijgen door gebruik te maken van de ROM-routine $FFF0. Deze routine kan de cursor op een willekeurige plaats op het scherm zetten. De regel waarop de cursor moet staan, moet in het X-register staan en de positie van de cursor op deze regel in het Y-register. Voor het aanroepen van deze routine moet ook nog de carry-flag gewist worden met CLC. Als voor het aanroepen de carry flag gezet is met SEC, dan wordt de cursor positie niet gezet maar gelezen. Het X-register bevat na de aanroep de regel waarop de cursor staat en het Y-register de positie op deze regel.

Cursor peeks & pokes

205:
Dit adres bepaalt de snelheid van het knipperen van de cursor. Standaard staat hier de waarde 20. Deze waarde wordt elke 1/60 seconde verlaagd. Zodra de waarde 0 is bereikt, wordt de cursor geïnverteerd en de waarde 20 weer teruggeplaatst in adres 205.

206:
Dit adres bevat de schermcode van het karakter dat onder de cursor zit.

214:
Hier wordt bijgehouden op welke regel de cursor staat. De waarde 0 betekent dat de cursor op de bovenste regel staat en de waarde 24 dat de cursor op de onderste regel staat.

647:
In dit adres staat de kleurcode van het karakter dat onder de cursor zit.

Border killer

Heeft u zich ook wel eens geërgerd aan de border om het scherm? Ja, dan zal na het lezen van dit stukje uw ergenis enigszins verminderd zijn. Het is namelijk mogelijk om met een klein machinetaal programmaatje de boven- en onder border uit te schakelen. De gedeeltes waar de border is verdwenen, hebben dan gewoon de kleur van het scherm. In die gedeeltes kan helaas geen tekst gezet worden, omdat het schermgeheugen maar 1000 bytes groot is. Het is echter wel mogelijk er sprites neer te zetten. Normaal zou een sprite met coördinaten 20,100 geheel achter de border verborgen zijn. Nu de border weg is, is deze sprite geheel zichtbaar.
Hoe is het mogelijk om de boven en onder border uit te schakelen? Er is immers geen register in de computer dat aangeeft of de border aan of uit is. Het programma werkt als volgt. Het beeldscherm is opgebouwd uit 312 horizontale rasterlijnen. Vijftig keer per seconde wordt het scherm van boven naar beneden opgebouwd. Het gedeelte binnen de borders begint bij rasterlijn 50 en eindigt bij rasterlijn 249. De onderborder begint dus op rasterlijn 250 en de bovenborder eindigt op lijn 49. Elke keer als de opbouw van het scherm bij lijn 250 is, wordt de border aangeschakeld. Deze blijft dan aan tot de opbouw bij rasterlijn 50 is; de border wordt dan weer uitgeschakeld. Normaal bestaat het scherm uit 25 tekstregels. We kunnen echter ook overschakelen op 24 regels door bit 3 van adres 53265 uit te schakelen. De boven- en onderborder worden dan iets breder. De onderborder begint dan op rasterlijn 247 i.p.v. 250. Door nu met een raster interrupt over te schakelen op 24 regels als de opbouw bij lijn 248 is en terug te schakelen op 25 regels bij 251, zal de border nooit worden aangezet. Dit is als volgt in te zien. Als de opbouw bij lijn 247 is, gebeurt er niets, want er zijn gewoon 25 re-

gels. Op lijn 248 wordt dan overgeschakeld op 24 regels, zodat de computer bij lijn 250 de border niet aanzet, omdat hij aanneemt dat dit al op lijn 247 is gebeurd. Et voila, de border wordt tijdens deze opbouw cyclus niet aangezet. Op lijn 251 wordt dan weer teruggeschakeld op 25 regels, zodat in de volgende opbouw cyclus weer dezelfde truc kan worden uitgehaald. Hieronder volgt de listing van het programma:

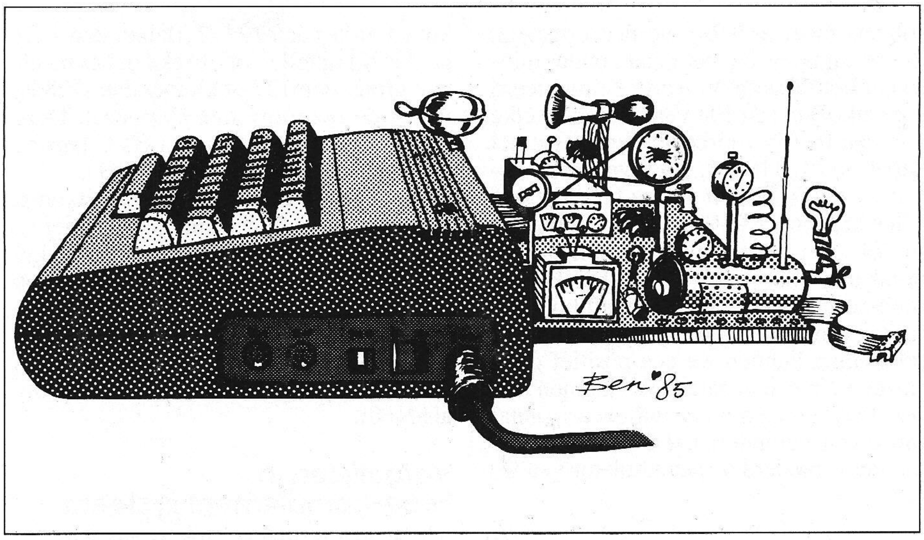
```
5 rem border killer
10 for i= 49152  to 49244
20 read a:s=s+a:poke i,a:next
30 if s11475 then print"fout
   in data!":end
40 sys 49152
100 data 120,169,44,141,20,3,
    169,192
110 data 141,21,3,169,1,141,
    26,208
120 data 169,249,141,18,208,
    173,17,208
130 data 41,127,141,17,208,
    169,127,141
140 data 13,220,169,0,141,
    255,63,141
150 data 92,192,88,96,169,1,
    141,25
160 data 208,173,92,192,48,
    19,173,17
170 data 208,41,247,141,17,
    208,206,92
180 data 192,169,252,141,18,
    208,76,129
190 data 234,173,17,208,9,8,
    141,17
200 data 208,238,92,192,169,
    248,141,18
210 data 208,76,49,234,0
```

Door in 16383 een waarde te poken, ontstaat er een streepjespatroon, op de plaats waar eerst de border zat. Als er een 1 in adres 16383 wordt gepoked, ontstaan er zwarte vertikale strepen. Met de waarde 255 wordt alles zwart. De kleur van de streepjes kan niet veranderd worden; deze is altijd zwart. Het volgende programma laat een van de effecten zien die met adres 16383 gemaakt kunnen worden. Voordat u het programma RUNt moet u natuurlijk wel de border uitschakelen.

```
10 poke 53280,14:poke 53281,6
20 for i=0 to 7
30 poke 16383,2~i
40 next
50 goto 10
```

Disk error

Als u probeert om een file, die niet bestaat, van disk te lezen, dan geeft de computer de foutmelding 'FILE NOT FOUND'. Bij andere fouten geeft de computer geen foutmelding; als er bijvoorbeeld geen disk in de diskdrive zit of als u een niet-bestaand commando aan de diskdrive geeft. In deze gevallen knippert alleen het error lampje op de diskdrive en



bij sommige fouten is er een flink geratel te horen. Als u wilt weten wat er precies fout is gegaan, moet u zelf het error kanaal van de diskdrive uitlezen. Dit kan in Basic als volgt:

```
10 open 15,8,15
20 input#15,en$,em$,t$,s$
30 print en$","em$","t$","s$
40 close 15
```

Dit programma drukt het foutnummer (en$), de foutmelding (em$) en de track (t$) en sector (s$), waar de fout optrad, af.

In Basic kunt u het error kanaal alleen maar met een programma uitlezen, omdat de INPUT instructie in regel 20 niet in direct mode kan worden uitgevoerd. De computer geeft dan de foutmelding 'ILLEGAL DIRECT ERROR'. Om het error kanaal uit te lezen, moet u dus steeds het programmaatje intikken. Dit is vervelend als u al een ander Basic programma in het geheugen heeft zitten. We hebben daarom een klein programma in machinetaal geschreven dat het error kanaal uitleest en de foutmelding afdrukt. Het enige wat u hoeft in te tikken om te kijken wat voor fout is opgetreden, is: SYS 49152. Hier volgt het programma.

```
5 rem error kanaal uitlezen
10 for i= 49152  to 49175
20 read a:s=s+a:poke i,a:next
30 if s3532 then print"fout
   in data!"
400 data
   169,8,32,180,255,169,111,3
410 data 150,255,32,165,255,
   32,210,255
420 data 201,13,208,246,32,
   171,255,96
```

Als u op adres 49152 al een ander programma hebt staan, dan kunt het begin- en eindadres van dit programma gewoon veranderen in regel 10.

Negatieve getallen

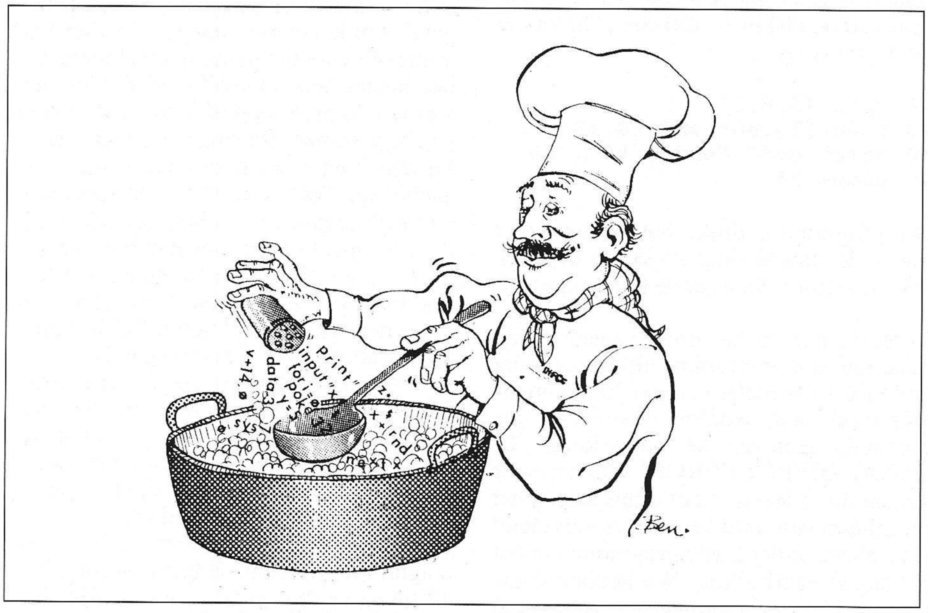
In Basic is het werken met negatieve getallen een eenvoudige zaak. Een negatief getal wordt gewoon aangeduid met een minteken voor het getal. In machinetaal is het helaas iets ingewikkelder. Getallen worden immers opgeslagen in de vorm van bitpatronen. De enige symbolen daarbij zijn 0 en 1. Er is dus geen minteken aanwezig. Toch kan er ook in machinetaal met negatieve getallen gewerkt worden. Normaal kan er met acht bits een getal van 0 tot 255 gerepresenteerd worden. Elk mogelijk bitpatroon komt dan overeen met een positief getal. Dit is prima als we alleen met positieve getallen werken. Als we echter ook met negatieve getallen willen werken, dan moeten een aantal bitpatronen worden gecorrespondeerd met negatieve getallen en een aantal met positieve. Hiervoor zijn een aantal verschillende systemen bedacht.

Het eenvoudigste systeem is het sign magnitude systeem. Hierbij wordt bit 7 als teken van het getal gebruikt. Als bit 7 een 0 bevat is het getal positief en als het bit een 1 bevat is het getal negatief. Het getal zelf staat in bits 0-6. In dit systeem wordt -1 dus als volgt gerepresenteerd: 10000001. Het probleem bij dit systeem is dat er nu twee bitpatronen voor het getal 0 zijn, namelijk 00000000 en 10000000. De computer maakt nu onderscheid tussen +0 en -0, terwijl deze gewoon gelijk zijn aan elkaar. Om dit probleem op te lossen is het twee-complement systeem bedacht.

Dit systeem gebruikt net als het sign magnitude systeem bit 7 als teken van het getal. Als het getal positief is (bit 7 heeft waarde 0), dan staat het getal weer in bits 0-6. De getallen 0 tot 127 hebben dus hetzelfde bitpatroon als wanneer we alleen met positieve getallen werken. Als het getal negatief is (bit 7 heeft waarde 1), kunnen we met een simpele berekening achterhalen wat het getal is. Het bitpatroon moet eerst geïnverteerd worden en ver-

volgens moet er 1 bij worden opgeteld. Het resultaat is dan het getal zonder minteken. Een bitpatroon wordt geïnverteerd door elke 0 in een 1 te veranderen en elke 1 in een 0. Als voorbeeld nemen we het bitpatroon 10110001. Na inverteren krijgen we 01001110 en als we hierbij 1 optellen krijgen we 01001111. Dit is het getal 79. Het oorspronkelijke bitpatroon stond dus voor -79. Met de berekening hebben het negatief getal eigenlijk veranderd in een positief. Met precies de zelfde berekening kunnen we een positief getal omzetten in een negatief. We kunnen met deze berekening dus een willekeurig getal van teken veranderen. De onderstaande routine veranderd de accumulator van teken.

```
eor #$ff    ;inverteer accu
clc
adc #$01    ;tel een op bij
    accu
```



Met het twee-complement systeem kunnen we met een byte getallen tussen -128 (10000000) en 127 (01111111) representeren. We kunnen natuurlijk ook met met twee bytes werken, zoals bij positieve getallen. Dan is niet bit 7 maar bit 15 het tekenbit. Er kunnen dan getallen tussen -32768 en 32767 worden gerepresenteerd. Optellen en aftrekken van getallen in het twee-complement systeem kan gewoon met ADC en SBC gedaan worden, waarbij voor optellen de carry flag gewist moet worden en voor aftrekken gezet moet worden. Als we alleen met positieve getallen werken, wordt de carry flag (C) gezet als de uitkomst van een optelling groter dan 255 is. En bij aftrekken wordt de carry flag gewist als de uitkomst kleiner dan 0 is. In het twee-complement systeem ontstaan er problemen als de uitkomst groter dan 127 of kleiner dan -128 is. Als bij optellen of aftrekken het resultaat groter dan 127 of kleiner dan -128 is, wordt de overflow flag (V) gezet. Deze flag kan getest worden met BVC (spring als V=0) en BVS (spring als V=1).
Er kan ook getest worden of de uitkomst van een berekening negatief of positief is. Als de uitkomst negatief is wordt de sign flag (N) gezet. Bij een positieve uitkomst wordt deze flag gewist. De waarde van de flag is dus gelijk aan de waarde van het tekenbit. Op deze flag kan getest worden met BMI (spring als N=1) en BPL (spring als N=0).

Vergelijken in twee-complement systeem

Als er alleen met positieve getallen gewerkt wordt, dan kan de accumulator bijvoorbeeld d.m.v. CMP #getal met een getal vergeleken worden. Als accu getal dan C=0. Als accu = getal, dan C=1. Bij getallen in het twee-complement systeem werkt dit helaas niet. Neem bijvoorbeeld accu=1 en getal=-1. Het bitpatroon voor -1 is 11111111. Dit interpreteert de computer echter niet als -1 maar als 255 en CMP #getal geeft dus C=0. In het twee complement-systeem is er een ingewikkelder algoritme nodig om twee getallen met elkaar te vergelijken. Hierbij is niet de C-flag maar zijn de N- en V-flag nodig. Daardoor kunnen we niet meer CMP gebruiken voor vergelijken, omdat deze instructie de V-flag niet zet bij een overflow. Om de accumulator met een ander getal te vergelijken trekken we dit getal van de accu af met SBC. Als nu (N=1 en V=0) of (N=0 en V=1) dan is accu getal. Anders is accu = getal. Doordat we het getal aftrekken van de accu, verandert de accu van waarde. Als deze waarde verder nog nodig is dan moet deze waarde zolang ergens bewaard worden. Het volgende programma vergelijkt de accu met een getal en drukt het resultaat af.

```
LABELS  MNEMONICS
    COMMENTAAR
        lda #1        ;Accu =
  1
        sta data      ;bewaar
  accu
        sec
        sbc #-1       ;getal =
  -1
        bmi min       ;spring
  als N=1
        bvs kleiner ;spring
  als V=1
        bvc groter  ;spring
  als V=0
min     bvc kleiner
        bvs groter
kleiner lda #       ;accu
    getal
        ldy #kl
        jsr $able
        jmp end
groter  lda #gg     ;accu =
    getal
        ldy #gg
        jsr $able
end     lda data      ;herstel
    accu
        rts
data    .byt 0
kl      .asc "accu  getal"
        .byt 0
gg      .asc "accu = getal"
        .byt 0
```

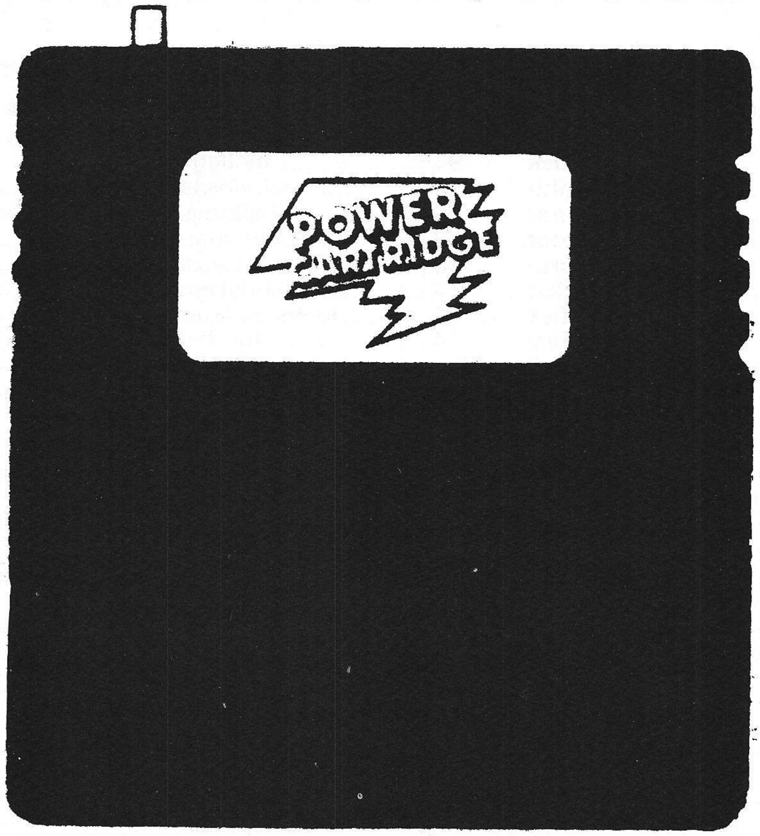
In regel 4 staat SBC #-1. Als uw assembler geen negatieve getallen aan kan, moet u dit veranderen in SBC #255. Deze twee instructies zijn aan elkaar gelijk, want -1 en 255 hebben hetzelfde bitpatroon. Dit bitpatroon is het enige wat van belang is voor de computer. Dat wij dit bitpatroon bijvoorbeeld met -1 of 255 associeren maakt voor de computer niets uit. Het is slechts een interpretatie die wij er aan geven.

Heeft u nog leuke tips of trucs, stuur deze dan op naar Commodore Info, Postbus 43048, 1009 ZA Amsterdam o.v.v. Tips & Trucs 64. Greetings from an unknown Art.

Michel de Boer & Hylke SprangerC

De Powercartridge

Bij de meeste fanatieke C64 gebruikers staat hij bekend als de rode duivel. Nu bedoelen we niet de beroemde Belgische voetballers, maar de rode, (bijna) alles kunnende Powercartridge. De leverancier, Kolff Computer Supplies (KCS), heeft de Powercartridge vernieuwd en de prijzen aanmerkelijk verlaagd. Reden voor ons om deze module weer eens opnieuw te bekijken.



De Powercartridge is een uitbreiding speciaal voor de Commodore 64. Deze wordt in de, hoe kan het ook anders, cartrigdepoort gestoken. Nu is er 16K rom geheugen beschikbaar voor de software, en deze staat geheel buiten het geheugen van de Commodore. Deze ruimte wordt zowel gebruikt voor uitbreiding van de kernal als voor een basic. Wij gebruiken als redactie reeds vele jaren de Powercartridge. Niet alleen bij het programmeerwerk, maar juist de snelheid waarmee wordt geladen, maakt het werken met de Commodore 64 veel aantrekkelijker. Dat de funktie, toetsen hierbij ook echt een funktie hebben gekregen, maakt de cartridge tot een onmisbaar stukje gereedschap. De Powercartridge is al vanaf 1985 op de markt. Sinds die periode zijn er de nodige veranderingen aangebracht. Aan de eerste Powercartridge hing een prijskaartje van *f* 139,--. Na een voorzichtige start bleek dit voor de echte 64-gebruikers geen enkel probleem. In 1988/89 waren er zoveel cartridges gemaakt dat de prijs een flink stuk kon dalen, er hoefde nog maar *f* 99,- voor worden neergeteld. We waren ervan overtuigd, dat dit ongeveer de bodemprijs zou zijn. Eind vorig jaar (1989) kregen we een brief van KCS, dat de prijs gehalveerd werd. De Powercartridge gaat nu over de toonbank voor *f* 49,--. Hoe kan dit? Normaal gesproken kan een prijs voor een goed lopend artikel niet ineens met 50% dalen. Het antwoord hierop was vrij simpel: de hardware is verandert. Dit heeft een aantal voordelen:

- Er kan sneller veel sneller en dus goedkoper worden geproduceerd.
- Deze cartridge geeft een betrouwbaardere werking op het langere termijn.
- Door een andere indeling op de printplaat te maken is het veel eenvoudiger om zelf aanpassingen aan de cartridge te maken. Denk hierbij aan een reset, zelfs een aan/uit schakelaar behoort nu tot de mogelijkheden.

Sinds 1985 zijn er meer dan 100.000 van deze Powercartrigdes verkocht. Dat een prijsverlaging aanslaat blijkt wel uit het feit dat er eind vorig jaar, na de prijsverlaging, ruim 4000 Powercartridges werden verkocht.

Toolkit kommando's

De toolkit moet worden gezien als een hulpmiddel bij het programmeren. Programmeren is vaak (eentonig) herhalen. De Powercartrigde is een onmisbare hulp om de omslachtige opdrachten flink te vereenvoudigen. We noemen de 'tools' met een korte omschrijving.

- AUTO Er wordt nu na een return een nieuw regelnummer op het scherm gezet, de sprong grootte kan worden ingesteld.
- COLOR Hiermee worden de achtergrond, voorgrond en de lettertekens van kleur veranderd.
- DEEK De inhoud van twee geheugenadressen kunnen tegelijk worden opgevraagd.
- DELETE Hiermee kan één of meerdere regels tegelijk worden verwijderd uit de listing
- DOKE Hiermee worden twee poke's in één keer uitgevoerd.
- DUMP Geeft een lijst op het scherm van alle enkelvoudige variabelen.
- FIND Op een eenvoudige manier kan hiermee worden gezocht naar een

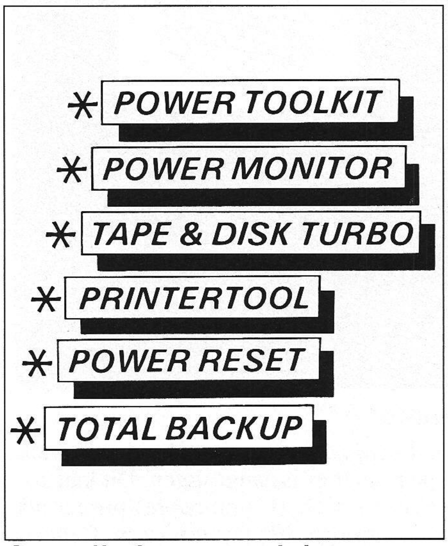
stuk tekst, of een variabel.

- HARDCAT De directory van een disk wordt op de printer afgedrukt.
- HARDCOPY Alles wat op het scherm staat wordt op de printer afgedrukt.
- HEX$ Is een funktie die van een getal de hexadecimale waarde uitrekend.
- INFO Toont alle beschikbare toolkit funkties op het scherm.
- KEY Alle gedefinieerde funktie toetsen worden getoond.
- MERGE Twee basic programma's kunnen tot één programma worden samengevoegd.
- PAUSE Last een pauze in in uw programma.
- PLIST De basiclist wordt op de printer afgedrukt.
- PSET Hiermee wordt de printer ingesteld.
- RENUM De listing wordt hernummerd, met een ingestelde waarde.
- REPEAT De toetsen van uw toetsenbord worden repeterend gemaakt.
- SAFE Schakeld de run/stop restore toets uit.
- TRACE Loopt het programma stap voor stap door.
- UNNEW Herstelt een programma dat per ongeluk is gewist.
- QUIT Hiermee wordt de Powercartridge uitgeschakeld.
- $ Laat een hexadecimale waarde uitvoeren, er hoeven geen moeilijke berekeningen meer worden uitgevoerd.

Disk kommando's

Een groot aantal programma's wordt door de Powercartridge sneller geladen. Voor programma's die een eigen (snel)laadt routine hebben, gaat dit meestal niet op. Een extra snelheidsvoordeel is dat bij dload, dsave en dverify geen device nummer behoeft te worden opgegeven. Bijvoorbeeld: DLOAD (sneller via de F7 toets) in de directory voor het programma van uw keuze zetten en het programma wordt zondermeer geladen. Een opgevraagde directory kan op het scherm worden gestopt door het indrukken van de spatiebalk.

- DLOAD Laadt (meestal) een programma sneller van de disk.
- DSAVE Schrijft een programma naar disk.
- DVERIFY Kontroleert of een programma goed is weggeschreven.
- DIR Toont de directory op het scherm zonder verlies van een eventueel in het geheugen aanwezig programma.
- DISK Een funktie voor het verkort weergeven van de disk kommandos, of de status op het beeldscherm te laten zien.
- DEVICE Met dit kommando wordt de 1541 diskdrive veranderd in devicenummer 9.
- MERGE Het koppelen van twee basic programma's tot één geheel.



Cassette kommando's

Programma's kunnen tot 10x zo snel worden geladen en gesaved en zelfs worden gekontroleerd (verify). Ook normaal laden en saven blijft tot de mogelijkheden behoren. Ook de meeste programma's die met een andere snellader zijn weggeschreven kunnen worden geladen.

- LOAD Door het geven van het devicenummer 2 wordt een programma, ook als het uit meerdere delen bestaat snel geladen. Dit geldt alleen als ze ook 'snel' zijn gesaved.
- SAVE Door het geven van het devicenummer 2 wordt een programma snel op de tape gezet. Nu kan het alleen met de snellader worden geladen.
- VERIFY Er kan met deze opdracht 10x sneller worden gekontroleerd of het programma goed naar de tape is weggeschreven.
- MERGE Het koppelen van twee basic programma's tot één geheel.
- AUDIO Het signaal dat op de cassette staat wordt via streepjes op het beeldscherm getoond. Tegelijk wordt via de speaker het signaal weergegeven. Het begin/eind van een programma kan zo makkelijk worden opgespoord.

Powermon monitor

Powermon is een machinetaal monitor/assembler. Deze stelt u in staat om gedeeltes van het geheugen op te vragen, hierin veranderingen aan te brengen en of te verplaatsen. We kunnen op verschillende manieren naar de monitor: via het resetmenu, door het indrukken van de F2 toets, het intikken van: MONITOR, het intikken van m[SHIFT O] of het intikken van :SYS$DE27. Programma's die zijn geschreven in machinetaal zijn honderden malen sneller dan het snelste basic programma. U kunt met deze monitor en de handleiding echter niet leren programmeren in machine taal. Hiervoor dient u een goed boek aan te schaffen, of bestudeer onze machinetaal cursus eens goed. Powermon staat in zijn geheel buiten het beschikbare geheugen van de Commodore. U kunt dus het gehele geheugengebied gebruiken. U heeft de beschikking over de volgende commando's:

- A Assembleren
- C Compare, vergelijken van geheugenadressen
- D Disassembleren, bekijken van geheugen adressen
- F Fill, het vullen van geheugengebieden met een waarde
- G Go run, het programma wordt uitgevoerd
- H Hunt, zoeken naar een bepaalde waarde
- I Interpret memory, het bekijken van het geheugen
- J Jump, ga naar een bepaald adres
- L Load, laden van tape of disk
- M Memory display, bekijken van het geheugen
- P Printen, gegevens naar de printer sturen
- R Register, toont de inhoud van de verschillende registers
- S Saven, een programma saven naar tape of disk
- T Transfer, verplaatsen van een stuk geheugen
- V Verify, kontroleren van een programma

- W Walk, stap voor stap doorlopen van het programma
- X Exit, verlaat de machinetaal monitor
- $ Dir, toont de directory op het beeldscherm

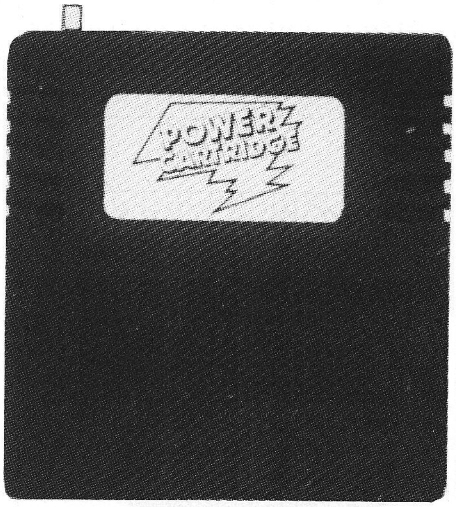
Power reset kommado's

Op de achterzijde van de Powercartridge bevind zich een resetknop. Door het indrukken van deze knop wordt de computer niet direct gereset. U komt nu in een menu. Voor de menu opties ziet U een handje. Deze is naar beneden te bewegen door het indrukken van de F7-toets. Naar boven gebeurt met de F1-toets.

- CONTINUE Het programma gaat weer verder van de plaats waar er op de resetknop is gedrukt, er is dus een pauze ingelast.
- BASIC Om terug te gaan naar de basic, met behoud van alle ingestelde waarden van de variabelen.
- RESET Dit is de normale reset funktie, alle variabelen gaan hierbij verloren.
- BACKUP DISK Hiermee kunt U het programma waar U mee bezig bent naar disk saven. Later kunt U dan weer verder gaan. Dit werkt niet bij alle programma's
- RESET ALL Reset alle programma's, ook die bij een normale reset verder gaan.
- BACKUP TAPE Werkt hetzelfde als backup disk alleen wordt het programma naar cassette geschreven.
- HARDCOPY Hiermee is het mogelijk om een beeldscherm afdruk op de printer te maken. Dit kan in groot of klein formaat.
- MONITOR Wordt gebruikt om naar de monitor te gaan.

Funktie toetsen.

De funktie toetsen hebben een voorgeprogrammeerde funktie. Snel laden van een programma kunnen we bijvoorbeeld doen door de F7 in te drukken, de directory verschijnt dan in het beeld. We zetten door middel van de cursor-toetsen de cursor op het gewenste programma en drukken op de F5 toets. Het programma wordt nu (snel) geladen. Runnen doen we door de F3 toets in te drukken. De listing wordt op het programma vertoond door het indrukken van de F1 toets. De listing scrollt van onder naar boven over het beeld. Door het indrukken van de spatiebalk stopt de listing. Verder kunnen we gaan door nogmaals de spatiebalk in te drukken. F2 brengt ons direkt naar de monitor. F4 is unnew, dus hiermee kunnen we een per ongeluk gewist programma probleemloos terug halen. Snelladen van tape kan door het indrukken van de F6 toets. Het eerste programma dat op de cassette staat wordt nu geladen. F8 is een toets die veel zal worden gebruikt. Er kunnen hiermee verkorte disk commando's worden gegeven. Het moeizaam openen en sluiten van kanalen behoort hiermee tot het verleden. Formatteren bijvoorbeeld kan nu gebeuren door het indrukken van de F7-toets gevolgd door N:disknaam,id



Printer

De Powercartridge controleert zelf welk type printer er is aangesloten. Dit kan zowel een seriële (Commodore) printer als een centronics (Pc printers) zijn. Centronics printers moeten via een speciale kabel worden aangesloten. Deze zijn gewoon in de handel, maar een handige computer freak kan deze zelf maken met behulp van het schema uit de handleiding. Door het PSET kommando kan de printer worden aangestuurd naar eigen behoefte en/of wensen. Alle PSET instellingen blijven ook na een reset gehandhaafd.

Uitbreidingen

Voorheen was het erg moeilijk om uitbreidingen, ook de eenvoudige, aan de Powercartridge te maken. Nu het produktie proces is veranderd, heeft ook de printplaat de nodige wijzigingen ondergaan. Het is nu relatief eenvoudig om een reset schakelaar te monteren. Een aan/uit schakelaar is ook een welkom hulpje op de cartridge. Bij de handleiding wordt zelfs een schema geleverd hoe het een en ander dient te worden aangesloten.

Conclusie

De Powercartridge is een handig, hulpmiddel voor de Commodore programmeur. Het kan veel tijd en ergernis besparen. De prijs/kwaliteits verhouding was al altijd goed. Nu is deze volledig omgeslagen: de kwaliteit ligt veel hoger dan de prijs. Zijn er dan helemaal geen nadelen te noemen? Ja natuurlijk wel; iets te wensen moet er altijd over blijven. De handleiding, is onhandig, het boekje moet steeds gedraaid worden. leest erg onhandig, en was voorheen veel beter. Gebruik liever het gehele boekje voor de Nederlandse handleiding, dit komt de leesbaarheid ten goede. De Powercartridge is eenvoudig te bedienen, dit 'ongemak' is dus alleen in het begin lastig. Nu is eigenlijk voor iedereen dit fraaie, nuttige, handige, rode doosje een betaalbare uitbreiding.

Fabrikant: Kolff Computer Supplies
Leverancier: Iedere goede computer handel
Prijs *f* 49,--

RBG

We hebben al enige keren tips cq uitbreidings programma's in ons blad vermeld. We hebben echter het idee, dat er nog veel meer gebruikers zijn die een eigen idee over de Powercartridge hebben. Er zijn gebruikers bekend die de nodige uitbreidingen voor dit stukje gereedschap hebben ontwikkeld. Wij zouden graag in één van onze volgende bladen een aantal van deze tips verzamelen. Dit kan niet zonder uw medewerking. Dus:

Heeft u een tip, misschien een uitbreiding voor het werken met de Powercartridge, die voor andere lezers ook van belang is
stuur deze tip/listing naar ons toe. Graag zoveel mogelijk informatie er bij. Soms is het ondoendelijk om een goed bedoelde tip te begrijpen. Eventuele listings meesturen op een diskette. We zullen u dan een nieuwe disk retourneren. Op de enveloppe vermeldt u dan LISTING POWER-CARTRIDGE !

Reacties naar:
SALA COMMUNICATIONS
t.a.v. R. Goudriaan
Postbus 43048
1009 ZA Amsterdam

Geos machinetaalcursus (deel 8)

De vorige keer werd het eerste deel van de labyrint-listing gegeven. In deze nieuwe aflevering van de Geos machinetaalcursus volgt de rest. Het programma wordt nu netjes aangekleed: er komt een grotere plattegrond, en de info en kaart-opties kunnen nu in werking gesteld worden. Het is de bedoeling dat de listings die deze keer geplaatst zijn, bij de vorige listings worden gevoegd. Sommige routines, die nu gelist worden, stonden in de vorige aflevering als dummy gelist. Deze dummy-labels moeten verwijderd worden.



Het is bij het publiceren van wat grotere programma's altijd een probleem hoever je moet gaan met het commentaar. Als je meer commentaar toevoegt dan eigenlijk noodzakelijk zou zijn, barst de listing al snel uit zijn voegen. Voor een tijdschrift zijn er daarom beperkingen aan de grootte van de programma's. In het geval van labyrint heb ik voor een zeer sobere aanpak wat betreft commentaar gekozen. Dat heeft natuurlijk ook nadelen. Er zullen ongetwijfeld lezers zijn die een programma als labyrint volledig willen begrijpen. Zonder commentaar als hulp is dat niet mogelijk. Ik zal daarom wat uitleg geven bij de procedures uit de nu geplaatste listing. Allereerst komen echter de routines 'draw' en 'directioninfo' aan de orde, hiervoor had ik vorige keer al wat extra tekst en uitleg beloofd.

Draw

Het plaatje van een kamer bestaat, zoals vorige aflevering al aan de orde is geweest, altijd uit 3 graphicsstrings: een voor up, een voor right en een voor left. Er is een jumptabel met pointers naar alle graphicsstrings. De jumptabel heet 'GraphTable' en stond in de vorige aflevering van deze machinetaalcursus. In de routine 'draw' worden er achtereenvolgens voor up right en left, pointers gezocht naar de goede graphicsstrings. Hiervoor geldt:

```
adres in jumptabel :=
    GraphTable + (fase * 12)
    + t * 4
```

Als er geen uitgang is, moet het adres bovendien nog met 2 opgehoogd worden. Hierin is t een teller die als er in de richting van up, right en left gekeken wordt resp. 0, 1 en 2 is. In de machinetaallisting is dit als volgt geimplementeerd:

-'fase' is de globale variabele waarin de fase staat. (nog2fase, nog1fase, keuzefase)

° a1H is vergelijkbaar met de genoemde variabele t.

° a2L bevat de richting waarvoor een graphicsstring gezocht moet worden. (up, right of left)

In het eerste gedeelte, tussen 'Draw:' en 'Drawlbl0:', wordt het tekenvenster zwart gemaakt. Dan wordt er overgeschakeld naar in de achtergrond tekenen ('LoadB dispBufferOn,64'). In het achtergrondscherm wordt het tekenvenster dan weer wit gemaakt, en hierin wordt later met behulp van de graphicsstrings het plaatje getekend.

Het tweede gedeelte, tussen 'Drawlbl0:' en 'Drawlbl1' houdt zich geheel bezig met het uitrekenen van de besproken formule.

In het laatste stuk 'Drawlbl2:' tot het einde, wordt het tekenvenster teruggekopieerd naar het voorgrondscherm met RecoverRectangle. Tevens wordt het aantal overgebeleven stappen met 1 verminderd, en het 'step' en 'compass' venster wordt overnieuw getekend. Mocht het aantal te nemen stappen op zijn, dan zorgt 'draw' ervoor dat er naar de routine 'verloren' gesprongen wordt.

Directioninfo

Deze routine heeft tot doel informatie te geven over het al dan niet aanwezig zijn van een uitgang in een bepaalde richting. Hiervoor krijgt directioninfo de volgende informatie:

° globale variabele 'room', met het kamernummer, waarin de speler zich bevindt.

° globale variabele 'direction', met de richting waarin de speler loopt.
° y, hierin wordt door de gebuiker van de procedure de richting waarin gekeken moet worden naar een uitgang, gezet. (relatief t.o.v. direction natuurlijk)

De routine rekent de volgende formule uit:

```
plaats in kaart := kaart +
    (room-1)*6 +
    (direction+y) AND #3
```

Als de gevonden plaats waarde 0 heeft (geen uitgang), dan wordt er 0 in x gezet, als de waarde 255 is (DE uitgang uit het doolhof), dan wordt er 2 in x gezet, en bij andere waarden krijgt x als inhoud 1. Precíese informatie over het formaat van de kaart stond in de vorige aflevering.

Drawsteps

Drawsteps is een hele simpele routine, die het steps-venster wist en het nieuwe aantal nog toegestane stappen erin zet. Dit gebeurt met de (hopelijk) bekende routine PutDecimal. Niet bekend? Let dan nu op: PutDecimal verwacht in r1H de x-coordinaat, in r1l de y- coordinaat, en in r0 het te printen getal. De a bevat informatie over de layout: bit 7 aan voor rechts uitvullen, uit voor links uitvullen. Bit 6 uit voor het printen van beginnullen aan het begin van het getal, aan om dit niet te doen.

Drawcompass

Deze routine tekent een pijl in het 'compass'-venster. De vier mogelijke pijlen (naar boven, onder, rechts en .. links) zijn alweer opgeslagen in graphicssstrings. Deze strings staan in de listing van deze keer en heten 'UpString', 'RightString', 'DownString' en 'LeftString'. Allereerst wordt natuurlijk de oude inhoud van het kompas-venster gewist. Daarna beperkt de routine zich ertoe de bij de in 'direction' opgeslagen richting horende string te graphicsstring-en.

DialogBoxes

Voor de volgende routines is het wenselijk dat U weet wat een DialogBox is, en hoe zo'n ding gemaakt wordt. Dit is in een eerdere aflevering van de machinetaalcursus al aan de orde geweest, maar omdat dit al een tijd terug is, zal ik er nog kort wat over zeggen. Een DialogBox (ook wel: DB) is een venster dat over het scherm wordt gezet, en dat de gebruiker een keuze laat maken, iets vertelt, of waarschuwt over de neveneffecten van een bepaalde handeling (etc). Een DB wordt op het scherm gebracht door de Geos-routine DoDlgBox aan te roepen. Geos zoekt dan de DB-tabel van de programmeur op. Deze staat als het goed is op het adres, waarnaar r0 wijst. De eerste byte van een de DB-tabel heeft bit 7 aan als de standaard afmetingen van een DB gebruikt worden. Als bit 7 uit is, dan volgen er 2 bytes (Y1 en Y2) en 2 words (X1 en X2) met het formaat van de DB. De rest van de eerste byte (bit 6 - bit 0) wordt gevuld met het patroon voor de schaduwrand onder de DB, patroon 0 betekent: geen schaduwrand. Vervolgens komen de DB- commando's. Ik zal alleen de DB-commando's 1-6, 11, 18 en 19 bespreken, omdat die in Labyrint gebruikt worden.

ICONS

De commando's 1 tot 6 lijken erg op elkaar en hebben ook bijna hetzelfde formaat: eerst het commando-nummer (1 tot 6) daarna een byte met de x-postitie in BYTES (per 8 puntjes dus) en de y- positie in pixels. Commando 1 tot en met 6 brengen respectievelijhk een 'OK', 'CANCEL', 'YES', 'NO', 'OPEN' en 'DISK' icon in de DB.

DBTXTSTR

Commando 11 is DBTXTSTR en print een tekst in de DB. Allereerst volgt weer het nummer (18) in een byte, dan de x-positie voor de tekst vanaf de linkerkant van de DB in pixels (ook een byte), en dan een byte met de y-positie in pixels. Tenslotte een pointer (een word dus) naar de string zelf.

DBUSRICON

Oftewel: DB-commando 18 zorgt ervoor, dat een door de gebruiker ontworpen icon op het scherm komt. Het formaat is precies dat van DBTXTSTR, met het verschil dat de pointer niet naar een string wijst, maar naar een zg. 'icon-infotabel'. Deze tabel bestaat uit een pointer naar het icon, dan 2 nullen, vervolgens een byte met de breedte in bytes en een byte met de hoogte in pixels (van het icon) en tenslotte een pointer naar de uitvoerroutine, als het icon met de muis aangeklikt wordt.

DB_USR_ROUT

Commando 19 tenslotte is een kort verhaal. Dit commando heet DB_USR_ROUT, en roept direct een user-routine aan. Het formaat is als volgt: eerst een byte met het nummer (19), en dan een pointer naar de user-routine.

Uit de DB kom je door een icon aan te klikken, dus door commando 1 tot 6 of 18. Het nummer van het DB commando, waardoor de gebruiker uit de DB ging, wordt in r0L gezet. Het programma gaat verder precies achter jsr DoDlgBox, en de programmeur moet zelf het scherm herstellen. De DB staat dan nog op het scherm. Dit was in razendsnelle vaart hoe een DB werkt. Nu terug naar het labyrintprogramma.

Gewonnen/verloren

Deze twee routines dienen om als de speler respectievelijk gewonnen of verloren heeft, daarvan melding te maken. Dit doen ze door een DialogBox op het scherm te brengen. Beide dialog-boxes lijken erg op elkaar: er is een OK-icon, waarmee de speler uit de dialog-box kan komen. Verder worden er gewoon een stel teksten in de DB gezet. De inforoutine doet eigenlijk precies hetzelfde: een DB met een OK-icon en een heleboel tekst. Alle DB-tabellen staan in de aanvulling voor de tabel-file. Alle te printen teksten in de tekstfile-aanvulling.

MapRoutine

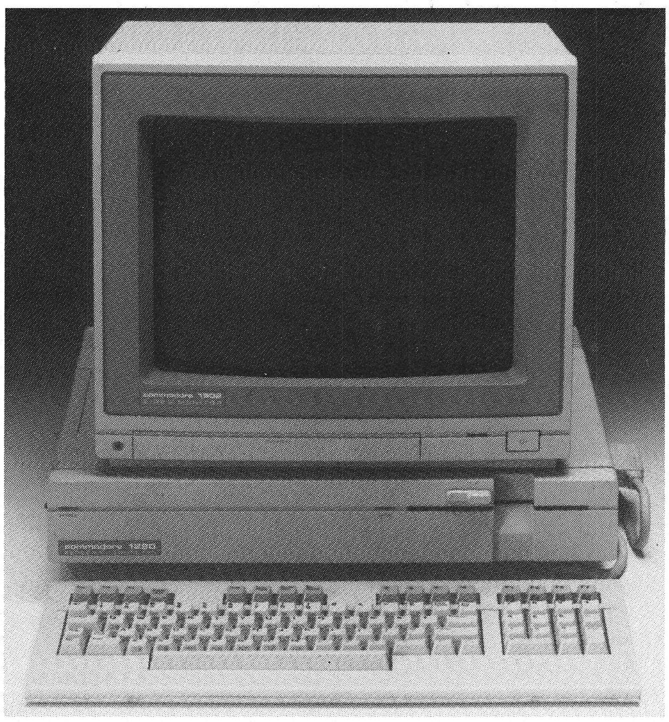
Deze routine heeft tot doel om eerst te controleren of de speler nog wel op de kaart mag kijken. Kan dit nog, dan wordt de mapteller met 1 verminderd, en wordt er een DialogBox, de MapDB, op het scherm gebracht. Deze MapDB heeft een OK-icon, en twee zelf gedefineerde icons. Dit zijn twee naar boven wijzende pijlen, een wijst naar de ingang van het doolhof, de andere naar de uitgang. Als het OK-icon aangeklikt wordt, dan wordt de DB beëindigd, de twee pijlen hebben geen functie voor het klikken met de muis. Tevens wordt in de MapDB de routine drawrooms aangeroepen. Deze routine is een nogal uitputtende aangelegenheid. Alle kamers worden afgegaan en als een rechthoekje in de kaart getekend op hun coördinaten. Deze coördinaten zijn per kamer in de kaart opgeslagen (zie deel 7). Tevens worden de aanwezige uitgangen naar andere kamers getekend. De kamer waar de speler in staat, wordt zwart ingekleurd. Al met al vergt dit nogal wat rekenwerk, heel erg moeilijk is het allemaal niet. Globaal werkt het als volgt:

Tussen 'Roomlbl0:' en 'Roomlbl6:' worden de coördinaten van de kamer omgerekend naar beeldschermcoordinaten voor de FrameRectangle. Hierbij is een kamer de rechthoek (48+X,18+Y,56+X,24+Y) met:

X := xco * 16 en Y := yco * 10 (xco en yco zijn de in de kaart opgeslagen coördinaten). De rest van 'DrawRooms' houdt zich bezig met het zetten van verbindingsssstreepjes bij de hierboven beschreven kamer als er uitgangen naar andere kamers zijn.

Quick/desktoproutine

Deze routines doen een jsr EnterDeskTop, maar dat is niet alles. Omdat er mensen zullen zijn, die met mijn programma 'QuickTop' werken, heb ik daarvoor een speciale uitgang gemaakt. Dat



kunt U zien als een soort van eigen produkt-support. Hoe dan ook, om de QuickTop voor de DeskTop uit te wisselen, moet er in de Geos- kernal een tekststring omgezet worden. Het vervelende is, dat voor de verschillende Geos-versies deze tekst op verschillende plaatsen staat. Maar daarover niet getreurd, want labyrint kijkt gewoon welke versie er gebruikt wordt. (Het versienummer staat altijd op $c00f) Voor Geos 1.x moet er 'quicktop' op $ccdb gezet worden, voor Geos 2.0 op $c3cf.

Netjes programmeren

In het begin van dit artikel heb ik de aandacht gevestigd op het feit, dat commentaar eigenlijk erg wenselijk is in machinetaalprogramma's. Tevens heb ik geprobeerd in de listing een beetje duidelijke namen aan te houden. Om de listing niet al te groot te maken, is hij bovendien opgedeeld in een aantal losse files (labyrintLst, tekstfile, tabelfile, kaartfile). Deze handelwijze kan als volgt beargumenteerd worden. Bij grotere machinetaalprogramma's is het van vitaal belang om het overzicht te bewaren. Tegelijk is het zo dat machinetaal niet de mogenlijkheden biedt, die programmeertalen zoals bijv. PASCAL (opsplitsen in procedures) of MODULA-2 (opsplitsen in procedures en het opdelen van programma's in losse modules) hebben. Wat de programmeur wel kan doen is netjes werken. Hieronder valt:

° constanten gebruiken;

° duidelijke (lange) namen met een grote psychologische afstand voor routines en constantes bedenken;

° het mooi houden van de listing.

Bij het laatste punt kan men denken aan een duidelijke layout en bijvoorbeeld het opdelen van de listing in sublistings. Kortom, een nette listing is een listing die een half jaar na het schrijven nog steeds begrijpelijk is voor de programmeur. Het moge duidelijk zijn, dat hierboven onder machinetaalprogrammeren, het programmeren met een assembler verstaan wordt. Het maken van een groot programma in een monitor, dus zonder labels en constanten, is natuurlijk knap, maar voor de rest alleen een hopeloze tijd- en moeiteverspilling. Tot slot nog een stel andere voorbeelden hoe het NIET moet:

° routines voor andere doeleinden gebruiken dan de naam van de routine aangeeft;

° tellers met bijvoorbeeld naam t, want wat telt die teller dan?;

° programmeer-trucjes gebruiken om ten koste van alles het programma nog net een regeltje korter of nog net een miliseconde sneller te maken;

bug.

Een interessant feit voor gebruikers van het GeoAssembler pakket is het volgende: ten eerste is het zo, dat van namen groter dan 8 letters de overblijvende letters afgehakt worden. Namen als bijvoorbeeld 'LabyrintLbl1' en 'LabyrintLbl0' zijn voor de computer gelijk aan elkaar. Hiervoor moeten 'nette' programmeurs dus oppassen, want duidelijke namen worden al gauw lang. Ten tweede zit er in de GeoAssembler een bug, want in geval van zo'n dubbel-label fout, construeert de GeoAssembler een error-file met een volledig foute VLIR-recordtabel, waardoor GeoWrite crasht als je zo'n file probeert in te lezen!

Tot slot

Alle nu gegegevn listings moeten bij die van vorige keer gevoegd worden. De listing met de programma-code (die begint met 'DrawSteps:') moet achteraan de LabyrintLst van vorige keer gevoegd worden. De labels zonder code die aan het einde van die listing stonden, moeten verwijderd worden. Bij alle andere listings van deze keer staat vermeld bij welke file ze toegevoegd moeten worden. Om het geval werkend te krijgen moet LabyrintLst weer door de GeoAssembler gehaald worden, en de labyrint.ink door de GeoLinker gehaald worden. De executeerbare file die dan op Uw disk opduikt heet dan 'labyrint.ex'. Volgende keer ben ik van plan om nog wat verder door te zagen over netjes programmeren. De machinetaal-cursus zal dan onder andere aandacht besteden aan het maken van macro's. Dit zijn vooraf gedefinieerde stukjes machinetaal, die overal waar de macro in de tekst voorkomt door de assembler in de machinetaallisting worden ingepast. Maar daarvover meer in de volgende aflevering.

Peter Boncz

```
DrawSteps:      lda #0
                jsr SetPattern
                jsr i_Rectangle
                .byte 37,55
                .word 264,304
                lda #1
                jsr SetPattern
                lda steps
                sta r0L
                lda steps+1
                sta r0H
                LoadB r1H,50
                LoadW r11,270
                lda #192
                jsr PutDecimal
                rts;

DrawCompass:    lda #0
                jsr SetPattern
                jsr i_Rectangle
                .byte 25,55
                .word 204,244
                lda direction
                and #3
                cmp #up
                beq Complbl0
                cmp #right
                beq Complbl1
                cmp #down
                beq Complbl2
                LoadW r0,LeftString
                jmp Complbl4
Complbl0:       LoadW r0,UpString
                jmp Complbl4
Complbl1:       LoadW r0,RightString
                jmp Complbl4
Complbl2:       LoadW r0,DownString
Comlblb4:       jsr GraphicsString
                rts;

Gewonnen:       jsr i_ImprintRectangle
                .byte 0,15
                .word 0,68
                LoadB running,FOUT
                LoadW r0,WinDB
                jsr DoDlgBox
                jsr i_RecoverRectangle
                .byte 0,199
                .word 0,319
                rts;

Verloren:       jsr i_ImprintRectangle
                .byte 0,15
                .word 0,68
                LoadB running,FOUT
                LoadW r0,VerliesDB
                jsr DoDlgBox
                jsr i_RecoverRectangle
                .byte 0,199
                .word 0,319
                rts;

InfoRoutine:    jsr GotoFirstMenu
                jsr i_ImprintRectangle
                .byte 0,15
                .word 0,68
                LoadW r0,InfoDB
                jsr DoDlgBox
                jsr i_RecoverRectangle
                .byte 0,199
                .word 0,319
                rts;

MapRoutine:     jsr GotoFirstMenu
                lda running
                cmp #FOUT
                beq Maplbl0
                lda mapteller
                cmp #0
                beq Maplbl0
                dec mapteller
                lda Kaart+4
                sta BeginX
                asl BeginX
                lda Kaart+10
                sta EindX
                asl EindX
                jsr i_ImprintRectangle
                .byte 0,15
                .word 0,68
                LoadW r0,MapDB
                jsr DoDlgBox
                jsr i_RecoverRectangle
                .byte 0,199
                .word 0,319
Maplbl0:        rts;

DrawRooms:      LoadW a0,Kaart
                LoadB a1L,0
Roomlbl0:       ldy #5
                lda (a0),y
                sta a2L
                asl a2L
                asl a2L
                lda (a0),y
                clc
                adc a2L
                sta a2L
                asl a2L
                dey
                lda (a0),y
                sta a2H
                asl a2H
                asl a2H
                asl a2H
                asl a2H
                LoadB r2L,18
                LoadB r2H,24
                LoadW r3,48
                LoadW r4,56
                lda a2L
                clc
                adc r2L
                sta r2L
                lda a2L
                adc r2H
                sta r2H
                lda a2H
                adc r3L
                sta r3L
                lda #0
                adc r3H
                sta r3H
                clc
                lda a2H
                adc r4L
                sta r4L
                lda #0
                adc r4H
                sta r4H
                sec
                lda room
                sbc a1L
                cmp #1
                bne Roomlbl6
                lda #1
                jsr SetPattern
                jsr Rectangle
Roomlbl6:       lda #255
                jsr FrameRectangle
                ldy #0
                lda (a0),y
                cmp #0
                beq Roomlbl1
                LoadB r3L,16
                LoadB r3H,18
                LoadW r4,52
                jsr SetVertical
                lda #255
                jsr VerticalLine
Roomlbl1:       ldy #1
                lda (a0),y
                cmp #0
                beq Roomlbl2
                LoadB r11L,21
                LoadW r3,56
                LoadW r4,60
                jsr SetHorizontal
                lda #255
                jsr HorizontalLine
Roomlbl2:       ldy #2
                lda (a0),y
                cmp #0
                beq Roomlbl3
                LoadB r3L,24
                LoadB r3H,26
                LoadW r4,52
                jsr SetVertical
                lda #255
                jsr VerticalLine
Roomlbl3:       ldy #3
                lda (a0),y
                cmp #0
                beq Roomlbl4
                LoadB r11L,21
                LoadW r3,44
                LoadW r4,48
                jsr SetHorizontal
                lda #255
                jsr HorizontalLine
Roomlbl4:       lda #6
                clc
                adc a0L
                sta a0L
                lda #0
                adc a0H
                sta a0H
                inc a1L
                lda a1L
                cmp #maxroom
                beeq Do_return
                jmp Roomlbl0
Do_return:      rts;

SetVertical:    clc
                lda a2H
                adc r4L
                sta r4L
                lda #0
                adc r4H
                sta r4H
                clc
                lda a2L
                adc r3L
                sta r3L
                lda a2L
                adc r3H
                sta r3H
                rts;

SetHorizontal:  clc
                lda a2H
                adc r3L
                sta r3L
                lda #0
                adc r3H
                sta r3H
                clc
                lda a2H
                adc r4L
                sta r4L
                lda #0
                adc r4H
                sta r4H
                clc
                lda a2L
                adc r11L
                sta r11L
                rts;

DeskTopRoutine:LoadW r0,DESKTOPTXT
                jsr InstallName
                jsr EnterDesktop
                rts;

QuickRoutine:   LoadW r0,QUICKTOPTXT
                jsr InstallName
                jsr EnterDeskTop
                rts;

InstallName:    lda $c00f
                cmp #$20
                beq Instlbl0

                LoadW r1,$ccdb
                jmp Instlbl1
Instlbl0:       LoadW r1,$c3cf
Instlbl1:       ldy #0
Instlbl2:       lda (r0),y
                sta (r1),y
                iny
                cpy #8
                bne Instlbl2
                rts;
```

```
;nieuwe kaartfile, moet op
;dezelfde disk als labyrintLst
;staan!
;let op!: verander in labyrintLst
; de constante maxroom in 197,
; en de constante maxsteps in 450

Kaart:
.byte 73,0,0,61,5,13
.byte 255,0,0,159,13,4
.byte 0,0,4,0,1,1
.byte 3,20,5,0,1,2
.byte 4,21,6,0,1,3
.byte 5,22,7,0,1,4
.byte 6,0,8,0,1,5
.byte 7,24,0,0,1,6
.byte 0,25,10,0,1,7
.byte 9,0,11,0,1,8
.byte 10,27,0,0,1,9
.byte 0,28,13,0,1,10
.byte 12,29,14,0,1,11
.byte 13,0,0,0,1,12
.byte 0,31,16,0,1,13
.byte 15,32,17,0,1,14
.byte 16,0,18,0,1,15
.byte 17,34,0,0,1,16
.byte 0,35,20,0,2,1
.byte 19,0,21,4,2,2
.byte 20,37,0,5,2,3
.byte 0,0,0,6,2,4
.byte 0,39,24,0,2,5
.byte 23,40,25,8,2,6
.byte 24,0,0,9,2,7
.byte 0,0,27,0,2,8
.byte 26,43,0,11,2,9
.byte 0,44,29,12,2,10
.byte 28,0,0,13,2,11
.byte 0,46,31,0,2,12
.byte 30,0,0,15,2,13
.byte 0,0,33,16,2,14
.byte 32,47,34,0,2,15
.byte 33,0,0,18,2,16
.byte 0,0,36,19,3,1
.byte 35,50,0,0,3,2
.byte 0,51,38,21,3,3
.byte 37,0,0,0,3,4
.byte 0,53,0,23,3,5
.byte 0,0,41,24,3,6
.byte 40,55,0,0,3,7
.byte 0,56,43,0,3,8
.byte 42,0,44,27,3,9
.byte 43,58,0,28,3,10
.byte 0,59,46,0,3,11
.byte 45,60,0,30,3,12
.byte 0,0,48,33,3,15
.byte 47,0,0,0,3,16
.byte 0,62,0,0,4,1
.byte 0,0,51,36,4,2
.byte 50,0,52,37,4,3
.byte 51,0,53,0,4,4
.byte 52,0,54,39,4,5
.byte 53,67,0,0,4,6
.byte 0,0,0,41,4,7
.byte 0,69,57,42,4,8
.byte 56,70,58,0,4,9
.byte 57,71,0,44,4,10
.byte 0,0,60,45,4,11
.byte 59,0,61,46,4,12
.byte 60,1,0,0,4,13
.byte 0,74,0,49,5,1
.byte 0,75,64,0,5,2
.byte 63,0,65,0,5,3
.byte 64,0,66,0,5,4
.byte 65,78,0,0,5,5
.byte 0,79,0,54,5,6
.byte 0,80,69,0,5,7
.byte 68,0,0,56,5,8
.byte 0,0,0,57,5,9
.byte 0,83,72,58,5,10
.byte 71,0,0,0,5,11
.byte 0,85,1,0,5,12
.byte 0,87,75,62,6,1
.byte 74,0,76,63,6,2
.byte 75,0,77,0,6,3
.byte 76,90,0,0,6,4
.byte 0,0,0,66,6,5
.byte 0,92,0,67,6,6
.byte 0,93,0,68,6,7
.byte 0,94,0,0,6,8
.byte 0,95,83,0,6,9
.byte 82,0,0,71,6,10
.byte 0,97,85,0,6,11
.byte 84,0,86,73,6,12
.byte 85,99,0,0,6,13
.byte 0,103,88,74,7,1
.byte 87,0,89,0,7,2
.byte 88,105,0,0,7,3
.byte 0,0,91,77,7,4
.byte 90,0,92,0,7,5
.byte 91,0,0,79,7,6
.byte 0,0,94,80,7,7
.byte 93,0,0,81,7,8
.byte 0,111,0,82,7,9
.byte 0,112,0,0,7,10
.byte 0,113,0,84,7,11
.byte 0,114,99,0,7,12
.byte 98,115,0,86,7,13
.byte 0,0,101,0,7,14
.byte 100,117,102,0,7,15
.byte 101,0,0,0,7,16
.byte 0,0,104,87,8,1
.byte 103,118,0,0,8,2
.byte 0,119,106,89,8,3
.byte 105,0,107,0,8,4
.byte 106,121,0,0,8,5
.byte 0,0,109,0,8,6
.byte 108,123,110,0,8,7
.byte 109,124,111,0,8,8
.byte 110,0,112,95,8,9
.byte 111,0,0,96,8,10
.byte 0,127,114,97,8,11
.byte 113,128,115,98,8,12
.byte 114,129,116,99,8,13
.byte 115,0,117,0,8,14
.byte 116,131,0,101,8,15
.byte 0,134,0,104,9,2
.byte 0,0,120,105,9,3
.byte 119,0,121,0,9,4
.byte 120,0,122,107,9,5
.byte 121,0,0,0,9,6
.byte 0,0,0,109,9,7
.byte 0,0,125,110,9,8
.byte 124,141,0,0,9,9
.byte 0,142,127,0,9,10
.byte 126,143,0,113,9,11
.byte 0,144,0,114,9,12
.byte 0,0,130,115,9,13
.byte 129,0,0,0,9,14
.byte 0,0,132,117,9,15
.byte 131,148,0,0,9,16
.byte 0,0,134,0,10,1
.byte 133,0,135,118,10,2
.byte 134,0,136,0,10,3
.byte 135,0,137,0,10,4
.byte 136,150,138,0,10,5
.byte 137,0,139,0,10,6
.byte 138,0,0,0,10,7
.byte 0,153,141,0,10,8
.byte 140,0,0,125,10,9
.byte 0,155,0,126,10,10
.byte 0,156,144,127,10,11
.byte 143,0,0,128,10,12
.byte 0,158,146,0,10,13
.byte 145,0,147,0,10,14
.byte 146,0,148,0,10,15
.byte 147,0,0,132,10,16
.byte 0,159,150,0,11,4
.byte 149,0,0,137,11,5
.byte 0,0,152,0,11,6
.byte 151,0,153,0,11,7
.byte 152,0,154,140,11,8
.byte 153,164,0,0,11,9
.byte 0,0,156,142,11,10
.byte 155,0,0,43,11,11
.byte 0,167,158,0,11,12
.byte 157,168,0,145,11,13
.byte 0,2,0,149,12,4
.byte 0,169,161,0,12,5
.byte 160,0,162,0,12,6
.byte 161,0,0,0,12,7
.byte 0,172,164,0,12,8
.byte 163,0,165,154,12,9
.byte 164,174,0,0,12,10
.byte 0,175,167,0,12,11
.byte 166,0,0,157,12,12
.byte 0,177,0,158,12,13
.byte 0,0,170,160,13,5
.byte 169,180,171,0,13,6
.byte 170,181,0,0,13,7
.byte 0,0,173,163,13,8
.byte 172,183,174,0,13,9
.byte 173,184,0,165,13,10
.byte 0,0,176,166,13,11
.byte 175,186,0,0,13,12
.byte 0,187,0,168,13,13
.byte 0,188,179,0,14,4
.byte 178,189,0,0,14,5
.byte 0,0,181,170,14,6
.byte 180,0,182,171,14,7
.byte 181,192,0,0,14,8
.byte 0,0,0,173,14,9
.byte 0,0,185,174,14,10
.byte 184,0,186,0,14,11
.byte 185,0,0,176,14,12
.byte 0,197,0,177,14,13
.byte 0,0,189,178,15,4
.byte 188,0,190,179,15,5
.byte 189,0,191,0,15,6
.byte 190,0,192,0,15,7
.byte 191,0,193,182,15,8
.byte 192,0,194,0,15,9
.byte 193,0,195,0,15,10
.byte 194,0,196,0,15,11
.byte 195,0,197,0,15,12
.byte 196,0,0,187,15,13
```

```
;aanvulling bij tabelfile:

WinDB:          .byte 1,25,90
                .word 60,304
                .byte 1,24,46
                .byte 11,2,11
                .word Win1Txt
                .byte 11,2,21
                .word Win2Txt
                .byte 11,2,31
                .word Win3Txt
                .byte 11,2,41
                .word Win4Txt
                .byte 11,2,51
                .word Win5Txt,0;

VerliesDB:      .byte 1,25,90
                .word 60,304
                .byte 1,24,46
                .byte 11,2,11
                .word Verlies1Txt
                .byte 11,2,21
                .word Verlies2Txt
                .byte 11,2,31
                .word Verlies3Txt
                .byte 11,2,41
                .word Verlies4Txt
                .byte 11,2,51
                .word Verlies5Txt,0;

MapDB:          .byte 1,20,190
                .word 47,304
                .byte 1,25,151
                .byte 18
BeginX:         .byte 0,137
                .word UpTable
                .byte 18
EindX:          .byte 0,9
                .word UpTable
                .byte 19
                .word DrawRooms
                .byte 0;

UpTable:        .word UpData,0
                .byte 3,24
                .word Do_return;

InfoDB:         .byte 0,1,199
                .word 37,308
                .byte 1,27,167
                .byte 11,2,11
                .word Info0Txt
                .byte 11,2,21
                .word Info1Txt
                .byte 11,2,31
                .word Info2Txt
                .byte 11,2,41
                .word Info3Txt
                .byte 11,2,51
                .word Info4xt
                .byte 11,2,61
                .word Info5xt
                .byte 11,2,71
                .word Info6Txt
                .byte 11,2,81
                .word Info7Txt
                .byte 11,2,91
                .word Info8Txt
                .byte 11,2,101
                .word Info9Txt
                .byte 11,2,111
                .word Info10Txt
                .byte 11,2,121
                .word Info11Txt
                .byte 11,2,131
                .word Info12Txt
                .byte 11,2,141
                .word Info13Txt
                .byte 11,2,151
                .word Info14Txt
                .byte 11,2,161
                .word Info15Txt
                .byte 11,2,171
                .word Info16Txt
                .byte 11,2,191
                .word Info17Txt,0;

UpString:       .byte 1,224,0,28,2
                .byte 217,0,52,2,231,0
                .byte 52,2,224,0,28,0
RightString:    .byte 1,237,0,40,2,211
                .byte 0,35,2,211,0,45
                .byte 2,237,0,40,0
LeftString:     .byte 1,211,0,40,2,237
                .byte 0,35,2,237,0,45
                .byte 2,211,0,40,0
DownString:     .byte 1,224,0,52,2,217
                .byte 0,28,2,231,0,28
                .byte 2,224,0,52,0
```

```
;aanvulling bij tekstfile:

Info0Txt:     .byte 24,"LABYRINT",27," gemaakt door ",24
              .byte "P. Boncz",27," in ",24,"1989",27,0
Info1Txt:     .byte "De bedoeling van het spel moet voor simpele C-64",0
Info2Txt:     .byte "bezitters zoals U en ik te begrijpen zijn: uit het",0
Info3Txt:     .byte "doolhof onsnappen! Dit dient te gebeuren binnen een",0
Info4Txt:     .byte "bepaald aantal stappen, dat in het 'steps' venster",0
Info5Txt:     .byte "staat. Wordt de limiet overschreden, dan volgt de",0
Info6Txt:     .byte "doodstraf. Bewegen door het doolhof gaat nogal",0
Info7Txt:     .byte "merkwaardig: lopen kan alleen met de naar boven",0
Info8Txt:     .byte "wijzende pijl, de andere pijlen zorgen er slechts",0
Info9Txt:     .byte "voor dat de looprichting draait. Er is een kompas",0
Info10Txt:    .byte "dat bijhoudt in welke richting er gelopen wordt.",0
Info11Txt:    .byte "Om het allemaal wat makkelijker te maken (een vlaag",0
Info12Txt:    .byte "van menslievendheid, U kent dat wel) heb ik besloten",0
Info13Txt:    .byte "om een extra optie, 'map' toe te voegen. Deze tovert",0
Info14Txt:    .byte "-of liever: tekent een kaart van het doolhof met Uw",0
Info15Txt:    .byte "huidige positie op het scherm. Deze 'map' optie",0
Info16Txt:    .byte "werkt echter maar viermaal per spel! ",0
Info17Txt:    .byte "Peter Boncz",0

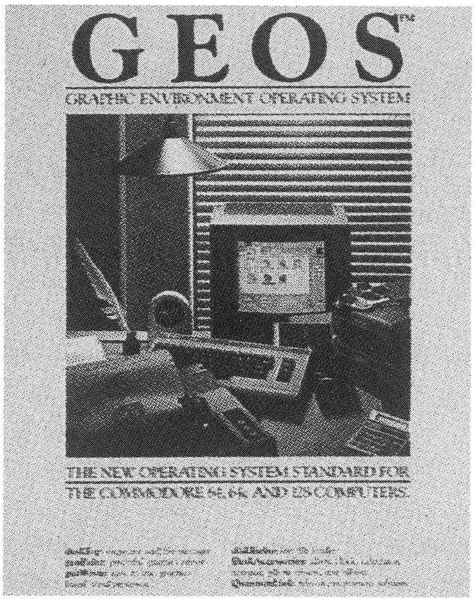
Win1Txt:      .byte 24,"Gefeliciteerd,",27
              .byte " U heeft het labyrint opgelost!",0
Win2Txt:      .byte "Wegens algemene bezuinigingen bij de Commodore",0
Win3Txt:      .byte "Info, kunnen wij Uw prestatie NIET met een prijs",0
Win4Txt:      .byte "belonen. Maar houd vol, de schouders eronder, en",0
Win5Txt:      .byte "de volgende keer eh.. eh.. weer geen prijs",0

Verlies1Txt: .byte 24,"system error near $0dd3:",27
             .byte " User brain not found.",0
Verlies2Txt: .byte "P.S.",0
Verlies3Txt: .byte "Het is hard, maar U heeft dit spel verloren.",0
Verlies4Txt: .byte "Klik nu het OK button om de diagnose van",0
Verlies5Txt: .byte "bovenaan te bevestigen.",0
```

GEOS INFO rubriek

Na het succes van de eerste GEOS-INFO Public Domain disk, werd het zolangzamerhand eens tijd voor een opvolger. De nieuwe disk bevat programma's uit de machinetaalcursus van Peter Boncz, en nog enkele ingezonden programma's. Onze oproep om zelfgemaakte GEOS- software in te zenden, was redelijk succesvol. Helaas waren een deel van de ingezonden programma's niet origineel (zonder kwade bedoelingen overigens), en in enkele andere zaten wat programmeertechnische gebreken. Er waren echter ook programma's bij die de toets der kritiek glansrijk konden doorstaan. Wij bedanken vooral Ronald de Man, maar natuurlijk ook alle andere inzenders, voor de gedane moeite. Nieuw in de opzet is dat er nu consequent bij ieder programma een '.doc' file zit met tekst en uitleg. Ik zal alle programma's even kort beschrijven:

Het spel Labyrint is onderwerp geweest van de GEOS-machinetaalcursus. Het is een driedimensionaal doolhof-spel. De bedoeling is, om binnen een bepaald aantal stappen uit het doolhof te ontsnappen. Daarbij mag een aantal keren op een kaart worden gekeken. Het lopen door het doolhof etc. is allemaal in GEOS-stijl uitgevoerd: icons en trekmenu's vieren hoogtij. Naast het programma-zelf is ook de source-code meegeleverd. Het doel hiervan is 2-ledig:
1) om duidelijk te maken hoe een GEOS-programma werkt;
2) om de gebruiker de gelegenheid te geven eventueel zelf aanpassingen te maken. Voorop staat dan wel, dat dit alleen voor eigen gebruik is, en dat steeds vermeld blijft waar het programma oorspronkelijk vandaan kwam en wie de oorspronkelijke auteur is.



Quicktop

De quicktop is een klein programmaaatje, dat de functies van de desktop ten dele overneemt. Met de Quicktop kunnen elementaire dingen, zoals het opstarten van een programma, of het kiezen van een printerdriver, worden gedaan. Het grote voordeel van de Quicktop is zijn kleine formaat. Dit is vooral van belang voor mensen met maar één diskdrive: op een werkdiskette hoeft dan geen desktop meer te staan, en de Quicktop neemt bijna geen ruimte in. Het verschil tussen deze Quicktop en de Quicktop die al op de vorige PD diskette stond, is dat deze ook vlekkeloos werkt met GEOS V2.0. Ook bij de Quicktop zijn de source-codes voor de GeoProgrammer gegeven.

Showlader

Dit programma een diashow over de diskette zelf. De files showdata 0 tot showdata 5 horen bij dit programma. Zij bevatten de dia's uit de show. Het programma showlader is een BASIC programma, dat ook deels in machinetaal is geschreven.

Galgje

Dit bekende oerhollandse vermaak is nu in een GEOS-programma omgezet. De computer kiest uit een lijst van 500 woorden, en voor U het weet bungelt U al aan de galg. Het opstarten van desk-accesoires blijft mogelijk gedurende het spel.

Vier op een Rij

De computer speelt met U, tegen zichzelf, of U speelt met een andere tegenstander. Er zijn drie moeilijkheidsniveaus. Tijdens het spelen kunt U de computer om advies vragen.

ArtImporter

Dit programma is vergelijkbaar met de welbekende 'Graphics Grabber', het is alleen veel goedkoper. Het leest plaatjes uit de PrintShop, PrintMaster en Newsroom bibliotheken en stopt ze in GEOS photo-albums of in losse 'scrap'-files voor GeoPaint.

IconChanger

Dit is de equivalent van de 'Icon Editor'. De IconChanger doet precies hetzelfde: alle icons van files op disk kunnen veranderd worden. Met precies dezelfde funkties (o.a. inverteren) als de officiële Icon Editor.

BESTEL NU DE GEOS-INFO Public Domain Disk 2:

F15,- overmaken op ..... onder vermelding van 'GEOS-INFO PD 2'

Checksum C-64

Syntax Checksum

Het overtikken van een listing kan een heel karwei zijn en als u een beetje normaal mens bent dan maakt u daarin beslist een aantal fouten. Nu is niets moeilijker om de fouten uit je eigen werk te halen. Al geruime tijd heeft Jan Bodzinga hiervoor een zgn. Checksum-programma geschreven. Om de vele nieuwe lezers van Commodore-info te helpen volgt hieronder nog een keer een volledige uitleg over de werking van dit programma, waarmee het, hoe vreemd dat misschien ook lijkt, echt mogelijk is om met behulp van dit programma de fouten in elke door ons geplaatste listing op te sporen.

Hiervoor gaat u als volgt te werk:

1. U tikt de listing heel zorgvuldig over en SAVEt hem voordat u het programma RUNt op een diskette of cassette.
2. U tikt het RUN commando in. Mocht het programma de boodschap 'FOUT in dataregels!' geven dan heeft u een fout bij het overtikken gemaakt. Herstel dan de fout en SAVE de verbeterde versie. Mocht het programma met de boodschap 'data is weggezet checksum testen met sys...'komen dan is tot dusver alles goed. Het programma is nu in een stukje machinetaalgeheugen gezet. Als u het NEW commando geeft blijft het toch in de computer staan.

Alle door ons geplaatste programma's zijn in Basic geschreven.

Als u een programma heeft overgetikt SAVE het eerst, mocht er iets mis gaan dan hoeft u niet de gehele listing opnieuw te gaan intikken. Als u nu een programma op fouten wilt gaan controleren dan kunt u dat in het geheugen laden(wel eerst het checksum programma hebben gerund). Vervolgens typt u zonder het programma te runnen de opdracht sys 49152(c-64) of sys 1536 (c-16 en plus/4)in.

Als alles goed is gegaan loopt er nu een rij regelnummers over het scherm met getallen erachter. Dezelfde lijst staat ook achter elk door ons geplaatste programma. Wijkt nu een nummer achter een regelnummer af van het nummer dat in het blad staat dan heeft u in die regel iets anders ingetikt dan er in het blad stond. U kunt de stroom getallen d.m.v. de RUN/STOP toets pauzeren en weer vervolgen met de F1 of F7 toets. Het is uitermate belangrijk dat u goed met dit programma overweg kunt en mocht u het niet goed werkend krijgen bel dan gerust even met onze listingservice telefoonlijn. (Maandag 17.00 - 21.00 uur. Telefoonnummer 02155-25162.)

De laatste tijd wordt er weer veel gebeld zodat U nogwel eens in gesprek krijgt, daarom houdt uw vraag kort, vermeld in welk blad het desbetreffende artikel stond. Heeft U veel vragen, of zis uw vraag erg uitgebreid, doe het dan schriftelijk, zodat we veel mensen op de maandag avond te woord kunnen staan.

```
1   rem ******************************
2   rem basic loader "SYNTAX.CHECKSUM"
3   rem na de commando's "run" en "new"
4   rem blijft dit programma in het ge-
5   rem heugen. laad het te testen pro-
6   rem gramma en tik daarna sys 49152.
7   rem *******************************
10  i=49152  :rem beginadres
20  reada:ifa<0then40:rem data ingelezen
30  pokei,a:i=i+1:b=b+a:goto20
40  if b<>16844thenprint"[SHIFT-CLR]fo
    ut[SPACE]in[SPACE]dataregels!":b=0
    :end
50  poke49184,148:poke49185,192
55  i=49300
60  read a: ifa<0then80
70  pokei,a:b=a+b:i=i+1:goto60
80  if b<>20068thenprint"[SHIFT-CLR]fo
    ut[SPACE]in[SPACE]dataregels![SPAC
    E](vanaf[SPACE]regel[SPACE]240)":b
    =0:end
90  print"data[SPACE]is[SPACE]weggezet"
95  print"checksum[SPACE]testen[SPACE]
    met[SPACE]sys49152"
100 data 165,43,166,44,133,163,134,164
    ,169,147
110 data 32,210,255,160,0,240,3,32,73,192
120 data32,73,192,208,1,96,32,225,255,208
130 data 3,76,116,164,32,81,192,32,73,192
140 data 240,12,201,32,240,247,24,101,
    167,133
150 data 167,76,37,192,166,167,169,0,1
    32,168
160 data 32,205,189,169,13,32,210,255,
    164,168
170 data 76,17,192,200,208,2,230,164,1
    77,163
180 data96,162,0,189,123,192,240,6,32,210
190 data 255,232,208,245,32,73,192,170
    ,32,73
200 data 192,132,168,32,205,189,162,3,
    169,32
210 data 32,210,255,202,208,250,169,0,
    133,167
220 data 164,168,96,82,69,71,69,76,32,0
230 data  -1
240 data 165,197,201,3,240,7,201,4,240
250 data 6,76,148,192,76,34,192,169
260 data 147,32,210,255,76,161,192
270 data -1
```

** EINDE LISTING checksum 64 **

```
REGEL   1   249          REGEL 100   183
REGEL   2   84           REGEL 110   158
REGEL   3   105          REGEL 120   232
REGEL   4   2            REGEL 130   183
REGEL   5   246          REGEL 140   96
REGEL   6   152          REGEL 150   96
REGEL   7   249          REGEL 160   127
REGEL  10   157          REGEL 170   71
REGEL  20   64           REGEL 180   223
REGEL  30   38           REGEL 190   73
REGEL  40   57           REGEL 200   79
REGEL  50   14           REGEL 210   109
REGEL  55   251          REGEL 220   106
REGEL  60   192          REGEL 230   225
REGEL  70   42           REGEL 240   16
REGEL  80   244          REGEL 250   163
REGEL  90   245          REGEL 260   92
REGEL  95   237          REGEL 270   22
```

PRINT OUT C-64 met o.a. paard

Shiftgame

Shiftgame is een spel dat geschreven is door Frank van Tiggelen uit Oud Beijerland. Het is een variatie op het wel bekende schuifspel. Shiftgame is een basic programma, 200 regels en maakt gebruik van twee kleine machinetaal routines. Dit is gedaan om de snelheid zo hoog mogelijk te houden. De eerste routine voegt de "Print At" funktie toe aan de bestaande basic. De tweede routine zorgt voor het aftellen van de seconden (dit gebeurt via de nmi-interrupt).

Extra informatie over de basic routines:

De routine voor de grote letters werkt als volgt: Alle 26 letters (inclusief de spatie) worden in 27 strings (L$(0) t/m L$(26)) ondergebracht. De strings bestaan uit drie rijen van drie grafische karakters die gescheiden worden door cursor-besturingstekens voor het juiste printen van de letters. De subroutine voor het printen geeft een string gecentreerd weer.

De woorden die in de strings (W$(0) t/m W$(26)) opgeslagen zijn, worden bij iedere beurt onderling verschoven ter bevordering van het toeval bij de woordkeuze.

Het tijdlimiet is aan het begin 30 seconden. Deze limiet wordt tot 20 tekens met twee verlaagd. Na deze 20 gaat het in stappen van 1.

De klok wordt door middel van poke nm,130 gestart en door middel van poke nm,130 gestopt. (nm=$DD0D)

```
10  rem *****************************
20  rem *     s h i f t g a m e     *
30  rem *                           *
40  rem * door: frank van tiggelen *
50  rem *                           *
60  rem *      oud-beijerland      *
70  rem *****************************
80  :
90  rem ** initialisatie **
100 :
110 poke53280,11:poke53281,11:diml$(26
    ),w$(26):gosub1340:fora=0to26:forb
    =0to2
120 reada$:l$(a)=l$(a)+a$+"[CRSR-DOWN]
    [3xCRSR-LEFT]":next:l$(a)=left$(l$
    (a),18)+"[3xCRSR-UP]":next:nm=56589
130 forx=0to170:ready:poke49152+x,y:n=
    n+y:next:ifn=19310thensys49152:got
    o390
140 print"fout[SPACE]in[SPACE]data[SPA
    CE]![SPACE](getallen)":stop
150 :
160 rem ** data voor letters **
170 :
180 data"UCI","[COM-Q]C[COM-W]","[COM-
    E][SPACE][COM-E]","[COM-A]I[SPACE]
    ","[COM-Q][COM-E]I","[COM-Z]CK","U
    CI","B[2xSPACE]","JCK","[COM-R]CI"
    ,"B[SPACE]B","[COM-E]CK"
190 data"[COM-A]C[COM-S]","[COM-Q]C[SP
    ACE]","[COM-Z]C[COM-X]","[COM-A]C[
    COM-S]","[COM-Q]C[SPACE]","[COM-E]
    [2xSPACE]","UCI","B[SPACE][COM-S]"
    ,"JC[COM-X]","[COM-R][SPACE][COM-R
    ]","[COM-Q]C[COM-W]","[COM-E][SPAC
    E][COM-E]"
200 data"[SPACE][COM-R][SPACE]","[SPAC
    E]B[SPACE]","[SPACE][COM-E][SPACE]
    ","[2xSPACE][COM-S]","[2xSPACE]B",
    "JCK","[COM-R][COM-A][SPACE]","[CO
    M Q][COM-E]I","[COM-E][SPACE][COM-
    Z]","[COM-R][2xSPACE]","B[2xSPACE]
    ","[COM-Z]C[COM-X]"
210 data"[3xCOM-R]","BBB","[COM-E][SPA
    CE][COM-E]","[COM-R]CI","B[SPACE]B
    ","[COM-E][SPACE][COM-E]","UCI","B
    [SPACE]B","JCK","[COM-R]CI","[COM-
    Q]CK","[COM-E][2xSPACE]"
220 data"UCI","B[SPACE]B","JC[SHIFT-+]
    ","[COM-R]CI","B[2xSPACE]","[COM-E
    ][2xSPACE]","UCI","JCI","JCK","[CO
    M A][COM-R][COM-S]","[SPACE]B[SPAC
    E]","[SPACE][COM-E][SPACE]"
230 data"[COM-R][SPACE][COM-R]","B[SPA
    CE]B","JCK","[COM-R][SPACE][COM-R]
    ","BUK","[COM-Z]K[SPACE]","[COM-R]
    [SPACE][COM-R]","BBB","[3xCOM-E]",
    "[COM-S][SPACE][COM-A]","JCI","[CO
    M X][SPACE][COM-Z]"
240 data"[COM-R][SPACE][COM-R]","J[COM
     R]K","[SPACE][COM-E][SPACE]","[CO
    M A]C[COM-S]","UCK","[COM-Z]C[COM-
    X]","[3xSPACE]","[3xSPACE]","[3xSP
    ACE]"
250 :
260 rem ** data voor print at & klok **
270 rem **            routine          **
280 :
290 data 169,61,141,10,3,169,192,141,1
    1,3,169,137,141,4,221,169,38,141,5,221
300 data 169,17,141,14,221,169,100,141
    ,6,221,169,0,141,7,221,169,81,141,15
310 data 221,169,127,141,13,221,169,11
    5,141,24,3,169,192,141,25,3,169,12
    7,141
320 data 13,221,96,169,0,133,13,32,115
    ,0,201,65,240,6,32,121,0,76,141,17
    4,160
330 data 1,177,122,201,84,208,242,32,1
    15,0,32,115,0,32,250,174,32,158,18
    3,138
340 data 72,32,241,183,104,168,24,32,2
    40,255,32,247,174,76,158,173,72,13
    8,72
350 data 152,72,173,13,221,16,11,174,7
    2,5,202,224,47,240,9,142,72,5,104,168
360 data 104,170,104,64,174,71,5,224,4
    8,240,12,202,169,57,141,72,5,142,71,5
370 data 76,136,192,230,2,169,127,141,
    13,221,76,136,192
380 :
390 forx=0to26:readw$(x):next:fori=0to
    int(rnd(0)*10):gosub1570:next
400 goto1760
410 :
420 rem ** woorden **
430 :
```

```
440  data radiospel,festivals,piraterij
     ,kwaliteit,shiftgame,vestibule,log
     istiek
450  data cartridge,damborden,croissant
     ,interrupt,korenveld,constante,oli
     fantje
460  data quasimodo,tydgebrek,vogelnest
     ,radiootje,huisafval,hoektafel,uit
     erlijk
470  data frikandel,zomertijd,revolutie
     ,categorie,antivries,ambulance
480  :
490  rem ** kies en verwar woord **
500  :
510  a=rnd(-ti):a=int(rnd(0)*26)+1:b=0:
     m=0:a$=""
520  ifm<62>50thengoto510
530  c=int(rnd(0)*9)+1:ford=0tob:ifal(d
     )=cthend=b:m=m+1:goto520
540  next:a$=a$+mid$(w$(a),c,1):al(b)=c
     :b=b+1:ifb<60>9then530
550  return
560  :
570  rem ** afdrukken woord met **
580  rem **     grote letters     **
590  :
600  printat(0,18)""
610  l=len(t$):ll=l*4:printtab(20-ll/2);
620  forx=1tol:printl$(asc(mid$(t$,x,1)
     )-65)"[CRSR-RIGHT]";:next:return
630  :
640  rem ** schermopbouw **
650  :
660  print"[SHIFT-CLR][CTRL-2]":t$="shi
     ftgame":gosub610
670  print"[CTRL-4][HOME][SPACE][COM-A]
     ";:forx=0to34:print"C";:next:print
     "[COM-S]":forx=0to2:print"[SPACE]B
     "tab(37);
680  print"B":next:print"[SPACE][COM-Q]
     ";:forx=0to34:print"C";:next:print
     "[COM-W]":forx=0to18
690  print"[SPACE]B"tab(37)"B":next:pri
     nt"[SPACE][COM-Z]";:forx=0to34:pri
     nt"C";:next:print"[COM-X][HOME]"
700  gosub710:goto740
710  printat(0,10)"[CTRL-4][SPACE][COM-
     Q]CCC[COM-R]CCC[COM-R]CCC[COM-R]CC
     C[COM-R]CCC[COM-R]CCC[COM-R]CCC[CO
     M R]CCC[COM-R]CCC[COM-W]"
720  forx=0to2:print"[SPACE]B[3xSPACE]B
     [3xSPACE]B[3xSPACE]B[3xSPACE]B[3xS
     PACE]B[3xSPACE]B[3xSPACE]B[3xSPACE
     ]B[3xSPACE]B":next
730  print"[SPACE][COM-Q]CCC[COM-E]CCC[
     COM-E]CCC[COM-E]CCC[COM-E]CCC[COM-
     E]CCC[COM-E]CCC[COM-E]CCC[COM-E]CC
     C[COM-W]":return
740  printat(0,6)"[2xCRSR-RIGHT][COM-A]
     CCCCCCCCC[COM-S][CRSR-UP][COM-A]CC
     CCCCCCCC[COM-S][CRSR-DOWN][COM-A]
     CCCCCCCCC[COM-S]"
750  print"[2xCRSR-RIGHT]B[CTRL-8]tijd[
     SPACE]over[CTRL-4]B[CRSR-UP]B[CTRL
      8][SPACE]woord[SPACE]:[3xSPACE][C
     TRL 4]B[CRSR-DOWN]B[2xSPACE][CTRL-
     8]score[CTRL-4][2xSPACE]B"
760  print"[2xCRSR-RIGHT]B[9xSPACE]B[CR
     SR-UP][COM-Q]CCCCCCCCCC[COM-W][CR
     SR-DOWN]B[9xSPACE]B"
770  print"[2xCRSR-RIGHT][COM-Z]CCCCCCC
     CC[COM-X][CRSR-UP]B[11xSPACE]B[CRS
     R-DOWN][COM-Z]CCCCCCCCC[COM-X]":pr
     inttab(13)"[CRSR-UP][COM-Z]CCCCCCC
     CCCC[COM-X]"
780  print"[5xCRSR-DOWN][3xCRSR-RIGHT][
     CTRL-8][COM-A]CCCCCCCCCCCCCCCCCCCC
     CCCCCCCCCCC[COM-S]":print"[3xCRSR-
     RIGHT]B[5xSPACE]hoogste[SPACE]s";
790  print"core[SPACE]:[11xSPACE]B"chr$
     (13)"[3xCRSR-RIGHT][COM-Z]CCCCCCCC
     CCCCCCCCCCCCCCCCCCCCCCC[COM-X]"
800  print"[CTRL-4][SPACE][COM-Q]CCCCCC
     CCCCCCCCCCCCCCCCCCCCCCCCCCCCC[COM-
     W][3xCRSR-DOWN]"
810  print"[CTRL-4][SPACE][COM-Q]CCCCCC
     CCCCCCCCCCCCCCCCCCCCCCCCCCCCC[COM-W]"
820  print"[2xCRSR-RIGHT][CTRL-2]door[S
     PACE]frank[SPACE]van[SPACE]tiggele
     n[2xSPACE](c)[SPACE]c-info":hi=100
     00:return
830  :
840  rem ** maak venster schoon **
850  :
860  forx=0to2:printat(2,19+x)"[CTRL-0]
     [35xSPACE]"
870  next:return
880  :
890  rem ** hoofdprogramma **
900  :
910  t$="druk[knop":print"[CTRL-1]";:go
     sub600:wait56320,16,16:gosub860:go
     sub1570
920  printat(11,20)"[CTRL-1]even[SPACE]
     geduld[SPACE]a.u.b.":gosub510:b$=w
     $(a):t$=a$:gosub860
930  printat(0,10)"[CTRL-2]":gosub610:p
     rintat(2,20)"[CTRL-1]spel[SPACE]is
     [SPACE]begonnen[SPACE]....";
940  print".[SPACE]tijd[SPACE]loopt[SPA
     CE]!":printat(15,8)b$:ifgm=0thensc
     =0:printat(28,8)"[CTRL-2]"sc
950  ifgm=0thentl=32:wr=0
960  i=1:a=3:k=0:print"[CTRL-9]";:gosub
     1250:tl=tl-1+(tl<62>20):wr=wr+1:print
     at(6,8)"[CTRL-0]"tl
970  printat(6,8)":":printat(24,16)hi:p
     rintat(22,6)wr
980  tl$=right$(str$(tl),2):printat(3,8
     )"[CTRL-9][CTRL-4]"tl$at(10,8)tl$"
     [CTRL-2]"
990  :
1000 rem ** schakel klok aan **
1010 :
1020 poke nm,130
1030 :
1040 rem ** spel **
1050 :
1060 ifpeek(1351)<62>48then1080
1070 ifpeek(1352)=48thengoto1610
1080 j=peek(56320):if(jand4)=0thengoto1120
1090 if(jand8)=0thengoto1140
1100 if(jand16)=16then1160
1110 i=0:goto1060
1120 ifa=0thengoto1100
1130 print"[CTRL-0]";:a=a+1:gosub1300:g
```

```
     osub1240:a=a-2:print"[CTRL-9]";:go
     sub1250:goto1100
1140 ifa=7thengoto1100
1150 print"[CTRL-0]";:gosub1300:gosub12
     40:a=a+1:print"[CTRL-9]";:gosub125
     0:goto1100
1160 ifi=1then1060
1170 pokez+1,250:pokez+5,4:pokez+6,1:po
     kez+4,129:pokez+4,128
1180 i=1:c$=mid$(a$,a+1,1):d$=mid$(a$,a
     +2,1):a$=left$(a$,a)+d$+c$+right$(
     a$,7-a)
1190 print"[CTRL-9]";:gosub1250:ifa$=b$
     thengoto1380
1200 goto1060
1210 :
1220 rem ** druk grote letters af **
1230 :
1240 print"[CTRL-2]"l$(asc(mid$(a$,a+1,
     1))-65);:return
1250 gosub1300:gosub1240:print"[CRSR-RI
     GHT]";:a=a+1:gosub1240
1260 a=a-1:return
1270 :
1280 rem ** positioneer letter **
1290 :
1300 printat(2+a*4,11)"";:return
1310 :
1320 rem ** reset sid **
1330 :
1340 z=54272:forx=0to24:pokez+x,0:next:
     pokez+24,15:return
1350 :
1360 rem ** woord ok **
1370 :
1380 poke nm,127 :rem ** klok uit **
1390 :
1400 gosub860:printat(0,18)"[CTRL-1]":t
     $="ok":gosub610
1410 :
1420 rem ** geluid **
1430 :
1440 forx=0to50step5:pokez+1,x:pokez+5,
     9:pokez+6,189:pokez+4,17:fory=0to30
1450 nexty,x:pokez+4,16:sc=sc+500+peek(
     1352)+peek(1351)*20+wr*25:rem * sc
     ore *
1460 printat(28,8)"[CTRL-2]"sc:gosub860
     :t$="volgend":printat(0,18)"[CTRL-
     1]":gosub610
1470 :
1480 pokez+5,0:pokez+6,0:gosub1530:t$="
     woord":gosub860:printat(0,18)"":go
     sub610
1490 gosub1530:gosub710:printat(15,8)"[
     9xSPACE]":gm=1:goto910
1500 :
1510 rem ** vertraginsloop **
1520 :
1530 forx=1to750:next:return
1540 :
1550 rem ** verschuif woorden **
1560 :
1570 a$=w$(0):forx=1to26:w$(x-1)=w$(x):
     next:w$(26)=a$:return
1580 :
1590 rem ** game over **
1600 :
1610 gosub860:printat(0,18)"[CTRL-1]"
1620 t$="game[over":gosub610:gosub1710:
     gosub1530:ifhi<60>scthenhi=sc
1630 gosub860
1640 printat(24,16)"[CTRL-2]"hi:t$="nie
     uw":printat(0,18)"[CTRL-1]":gosub6
     10:gosub1530:gosub860
1650 t$="spel":printat(0,18)"":gosub610
     :gosub1530:gm=0:gosub860
1660 printat(28,8)"[7xSPACE]"at(15,8)"[
     9xSPACE]"at(23,6)"[2xSPACE]":gosub710
1670 printat(3,8)"[9xSPACE]":goto910
1680 :
1690 rem ** geluid **
1700 :
1710 forx=75to1step-1:pokez+1,x:pokez+5
     ,9:pokez+6,0:pokez+4,17:pokez+4,16
1720 next:return
1730 :
1740 rem ** titelscherm **
1750 :
1760 print"[SHIFT-CLR][CTRL-2]":t$="shi
     ftgame":gosub610:print"[4xCRSR-DOW
     N][CTRL-4]":t$="door":gosub610
1770 print"[4xCRSR-DOWN][CTRL-8]":t$="f
     rank":gosub610:print"[3xCRSR-DOWN]
     ":t$="van":gosub610:print"[3xCRSR-
     DOWN]"
1780 t$="tiggelen":gosub610:printat(8,2
     3)"[CTRL-1](c)[SPACE]commodore[SPA
     CE]info[2xSPACE]1989"
1790 printat(4,24)"druk[SPACE]op[SPACE]
     de[SPACE]vuurknop[SPACE](joystick[
     SPACE]2)[HOME]":wait56320,16,16
1800 :
1810 rem ** instrukties **
1820 :
1830 print"[SHIFT-CLR][CTRL-2]":t$="shi
     ftgame":gosub610:forx=6to20:readt$
     :l=20-len(t$)/2
1840 printat(l,x)t$:next:wait56320,16,1
     6:gosub660:goto910
1850 data
    "<62><62><62><622[SPACE]instrukties[SP
    ACE]<60><60><60><60>"
1860 data "[2xSPACE]"
1870 data "het[SPACE]is[SPACE]de[SPACE]
     bedoeling[SPACE]van[SPACE]dit[SPAC
     E]spel[SPACE]om"
1880 data "zoveel[SPACE]mogelijk[SPACE]
     woorden[SPACE]in[SPACE]de[SPACE]goede"
1890 data "volgorde[SPACE]te[SPACE]zett
     en[SPACE]voordat"
1900 data "de[SPACE]tijd[SPACE]verstrijkt"
1910 data "er[SPACE]kunnen[SPACE]telken
     s[SPACE]2[SPACE]letters[SPACE]verw
     isseld"
1920 data "worden[SPACE]d.m.v[SPACE]een
     [SPACE]druk[SPACE]op[SPACE]de[SPAC
     E]vuurknop"
1930 data "door[SPACE]de[SPACE]joystick
     [SPACE]naar[SPACE]links[SPACE]en[S
     PACE]rechts"
1940 data "te[SPACE]bewegen[SPACE]kiest
     [SPACE]men[SPACE]de[SPACE]letters[
     SPACE]die"
1950 data "verwisseld[SPACE]moeten[SPAC
     E]worden"
```

```
1960 data "hoe[SPACE]meer[SPACE]woorden
     [SPACE]u[SPACE]haalt[SPACE],[SPACE
     ]hoe[SPACE]korter"
1970 data "de[SPACE]tijd[SPACE]voor[SPA
     CE]het[SPACE]volgende[SPACE]woord[
     SPACE]wordt"
1980 data "[2xSPACE]"
1990 data "druk[SPACE]op[SPACE]vuurknop"
2000 :
```

** EINDE LISTING shift-game **

Checksum Shift-game

REGEL 10	39	REGEL 700	131	REGEL 1390	58
REGEL 20	123	REGEL 710	158	REGEL 1400	29
REGEL 30	227	REGEL 720	248	REGEL 1410	58
REGEL 40	247	REGEL 730	28	REGEL 1420	241
REGEL 50	227	REGEL 740	98	REGEL 1430	58
REGEL 60	200	REGEL 750	164	REGEL 1440	14
REGEL 70	39	REGEL 760	128	REGEL 1450	110
REGEL 80	58	REGEL 770	106	REGEL 1460	248
REGEL 90	0	REGEL 780	249	REGEL 1470	58
REGEL 100	58	REGEL 790	32	REGEL 1480	219
REGEL 110	0	REGEL 800	182	REGEL 1490	138
REGEL 120	16	REGEL 810	131	REGEL 1500	58
REGEL 130	33	REGEL 820	92	REGEL 1510	118
REGEL 140	84	REGEL 830	58	REGEL 1520	58
REGEL 150	58	REGEL 840	66	REGEL 1530	128
REGEL 160	186	REGEL 850	58	REGEL 1540	58
REGEL 170	58	REGEL 860	234	REGEL 1550	4
REGEL 180	208	REGEL 870	74	REGEL 1560	58
REGEL 190	149	REGEL 880	58	REGEL 1570	46
REGEL 200	10	REGEL 890	77	REGEL 1580	58
REGEL 210	222	REGEL 900	58	REGEL 1590	141
REGEL 220	39	REGEL 910	97	REGEL 1600	58
REGEL 230	240	REGEL 920	114	REGEL 1610	125
REGEL 240	84	REGEL 930	63	REGEL 1620	188
REGEL 250	58	REGEL 940	142	REGEL 1630	43
REGEL 260	16	REGEL 950	36	REGEL 1640	39
REGEL 270	93	REGEL 960	126	REGEL 1650	103
REGEL 280	58	REGEL 970	2	REGEL 1660	213
REGEL 290	100	REGEL 980	100	REGEL 1670	183
REGEL 300	252	REGEL 990	58	REGEL 1680	58
REGEL 310	155	REGEL 1000	51	REGEL 1690	241
REGEL 320	123	REGEL 1010	58	REGEL 1700	58
REGEL 330	136	REGEL 1020	242	REGEL 1710	234
REGEL 340	127	REGEL 1030	58	REGEL 1720	74
REGEL 350	46	REGEL 1040	107	REGEL 1730	58
REGEL 360	49	REGEL 1050	58	REGEL 1740	123
REGEL 370	105	REGEL 1060	245	REGEL 1750	58
REGEL 380	58	REGEL 1070	127	REGEL 1760	71
REGEL 390	145	REGEL 1080	40	REGEL 1770	182
REGEL 400	87	REGEL 1090	229	REGEL 1780	177
REGEL 410	58	REGEL 1100	196	REGEL 1790	105
REGEL 420	85	REGEL 1110	181	REGEL 1800	58
REGEL 430	58	REGEL 1120	160	REGEL 1810	156
REGEL 440	45	REGEL 1130	207	REGEL 1820	58
REGEL 450	36	REGEL 1140	167	REGEL 1830	253
REGEL 460	31	REGEL 1150	132	REGEL 1840	53
REGEL 470	54	REGEL 1160	37	REGEL 1850	20
REGEL 480	58	REGEL 1170	2	REGEL 1860	199
REGEL 490	88	REGEL 1180	245	REGEL 1870	236
REGEL 500	58	REGEL 1190	247	REGEL 1880	144
REGEL 510	1	REGEL 1200	80	REGEL 1890	187
REGEL 520	180	REGEL 1210	58	REGEL 1900	147
REGEL 530	52	REGEL 1220	152	REGEL 1910	160
REGEL 540	162	REGEL 1230	58	REGEL 1920	153
REGEL 550	142	REGEL 1240	248	REGEL 1930	243
REGEL 560	58	REGEL 1250	37	REGEL 1940	59
REGEL 570	67	REGEL 1260	216	REGEL 1950	102
REGEL 580	219	REGEL 1270	58	REGEL 1960	136
REGEL 590	58	REGEL 1280	88	REGEL 1970	17
REGEL 600	136	REGEL 1290	58	REGEL 1980	199
REGEL 610	152	REGEL 1300	81	REGEL 1990	38
REGEL 620	1	REGEL 1310	58	REGEL 2000	58
REGEL 630	58	REGEL 1320	154		
REGEL 640	213	REGEL 1330	58		
REGEL 650	58	REGEL 1340	233		
REGEL 660	19	REGEL 1350	58		
REGEL 670	37	REGEL 1360	92		
REGEL 680	245	REGEL 1370	58		
REGEL 690	78	REGEL 1380	140		

Paard op schaakveld

J.D. Rienstra uit Sneek heeft voor ons een programma geschreven dat gebruik maakt van een schaakbord. Het enigste stuk dat nu wordt gebruikt is het paard. Het paard mag verplaatst worden op de bekende schaak manier. Voor de niet schakers onder ons, dit is twee velden naar links, rechts, onder of boven en daarna 1 veld opzij. Dit mag ook omgekeerd gebeuren. Zie de tekening, hierop is aangegeven welke velden kunnen worden gebruikt uit het positie van het paard.
De opdracht is nu dat elk veld één keer moet worden gebruikt en **niet vaker**. Verdere aanwijzinggen worden in het programma gegeven.

```
10   rem initialiseren
20   dima$(200):dimc$(200):dime$(200):d
     imf$(200):dimb$(100):v=53248
30   poke53280,0:poke53281,0:print"[SHI
     FT CLR]"
40   pokev+21,3:poke2040,13:poke2041,13
     :pokev+28,3:pokev+37,0:pokev+38,1
50   pokev+28,3
60   pokev,200:pokev+1,210:pokev+2,100:
     pokev+3,210
70   pokev+29,3
80   pokev+23,3
90   gosub1740:gosub1940
100  gosub1740
110  print"[CRSR-DOWN][CRSR-RIGHT][CTRL
      2][CRSR-RIGHT][CTRL-9][CRSR-DOWN]
     [SPACE]het[SPACE]paard[SPACE]op[SP
     ACE]het[SPACE]schaakbord"
120  print"[2xCRSR-DOWN][2xCRSR-RIGHT]h
     et[SPACE]paard[SPACE]moet[SPACE]in
     [SPACE]een[SPACE]vloeiende[SPACE]l
     ijn"
130  print"[2xCRSR-RIGHT]over[SPACE]het
     [SPACE]bord[SPACE]bewegen,op[SPACE
     ]de[SPACE]normale[5xSPACE]schaak[S
     PACE]manier."
140  print"[2xCRSR-RIGHT]elk[SPACE]veld
     [SPACE]moet[SPACE]een[SPACE]keer[S
     PACE]worden[SPACE]aange-[4xSPACE]d
     aan[SPACE]en[SPACE]niet[SPACE]vake
     r" "
150  print"[5xCRSR-RIGHT][SPACE][7xCRSR
     -DOWN]toets[SPACE]voor[SPACE]vervo
     lg"
160  get h$:if h$=""then 160
170  gosub1740
180  print"[2xCRSR-RIGHT][SPACE][2xCRSR
     -DOWN]het[SPACE]programma[SPACE]co
     ntroleert[SPACE]elke[SPACE]zet."
190  print"[2xCRSR-RIGHT]of[SPACE]het[S
     PACE]paard[SPACE]de[SPACE]juiste[S
     PACE]beweging[SPACE]maak"
200  print"[2xCRSR-RIGHT]zoals[SPACE]bi
     j[SPACE]gewoon[SPACE]schaken"
210  print"[2xCRSR-RIGHT]en[SPACE]of[SP
     ACE]de[SPACE]plaats[SPACE]al[SPACE
     ]eerder[SPACE]bezet"
220  print"[2xCRSR-RIGHT]is[SPACE]gewee
     st"
230  print"[2xCRSR-RIGHT][7xCRSR-DOWN]t
```

```
    oets[SPACE]voor[SPACE]vervolg"
240 get h$:if h$=""then240
250 gosub1740
260 print"[2xCRSR-RIGHT][2xCRSR-DOWN]a
    ls[SPACE]je[SPACE]denkt[SPACE]klaa
    e[SPACE]te[SPACE]zin[SPACE]of[SPAC
    E]wilt"
270 print"[2xCRSR-RIGHT]stoppen[SPACE]
    druk[SPACE]dan[SPACE]op[SPACE][CTR
    L 9]e[CTRL-0]"
280 print"[3xCRSR-DOWN][3xCRSR-RIGHT]j
    oystick[SPACE]in[SPACE]poort[SPACE
    ]1"
290 print"[CRSR-DOWN][3xCRSR-RIGHT]wil
    [SPACE]je[SPACE]een[SPACE]demonstr
    atie[SPACE]druk[SPACE]dan"
300 print"[3xCRSR-RIGHT]op[SPACE][CTRL
     9]d[CTRL-0][SPACE]en[SPACE]volg[S
    PACE]de[SPACE]aanwijzingen."
310 print"[5xCRSR-RIGHT][SPACE][6xCRSR
    -DOWN]toets[SPACE]voor[SPACE]start
    "
320 geth$:if h$=""then 320
330 pokev+29,0:pokev+23,0
340 pokev,0:pokev+1,0:pokev+2,0:pokev+
    3,0:pokev+28,0
350 tt=1:zet=1:m=0:oo=0
360 if h$="d" then dem= 1
370 tt=1:zet=1:m=0
380 rr=1:v=1:w=1
390 poke53280,0:poke53281,0
400 print"[SHIFT-CLR]":rem veld rechts
     opzetten
410 a=1026:gosub2530
420 poke1066,129:poke1067,184:poke1068
    ,160:poke1069,2:poke1070,56:poke10
    71,32
430 poke1072,131:poke1073,184:poke1074
    ,160:poke1075,4:poke1076,56:poke10
    77,32
440 poke1078,133:poke1079,184:poke1080
    ,160:poke1081,6:poke1082,56:poke10
    83,32
450 poke1084,135:poke1085,184:poke1086
    ,160:poke1087,8:poke1088,56:poke10
    89,32
460 a=1106:gosub2530
470 a=1149:gosub2530
480 poke1186,1:poke1187,55:poke1188,32
    :poke1189,130:poke1190,183:poke119
    1,160
490 poke1192,3:poke1193,55:poke1194,32
    :poke1195,132:poke1196,183:poke119
    7,160
500 poke1198,5:poke1199,55:poke1200,32
    :poke1201,134:poke1202,183:poke120
    3,160
510 poke1204,7:poke1205,55:poke1206,32
    :poke1207,136:poke1208,183:poke120
    9,160
520 a=1229:gosub2530
530 a=1266:gosub2530
540 poke1306,129:poke1307,182:poke1308
    ,160:poke1309,2:poke1310,54:poke13
    11,32
550 poke1312,131:poke1313,182:poke1314
    ,160:poke1315,4:poke1316,54:poke13
    17,32
560 poke1318,133:poke1319,182:poke1320
    ,160:poke1321,6:poke1322,54:poke13
    23,32
570 poke1324,135:poke1325,182:poke1326
    ,160:poke1327,8:poke1328,54:poke13
    29,32
580 a=1346:gosub2530
590 a=1389:gosub2530
600 poke1426,1:poke1427,53:poke1428,32
    :poke1429,130:poke1430,181:poke143
    1,160
610 poke1432,3:poke1433,53:poke1434,32
    :poke1435,132:poke1436,181:poke143
    7,160
620 poke1438,5:poke1439,53:poke1440,32
    :poke1441,134:poke1442,181:poke144
    3,160
630 poke1444,7:poke1445,53:poke1446,32
    :poke1447,136:poke1448,181:poke144
    9,160
640 a=1469:gosub2530
650 a=1506:gosub2530
660 poke1546,129:poke1547,180:poke1548
    ,160:poke1549,2:poke1550,52:poke15
    51,32
670 poke1552,131:poke1553,180:poke1554
    ,160:poke1555,4:poke1556,52:poke15
    57,32
680 poke1558,133:poke1559,180:poke1560
    ,160:poke1561,6:poke1562,52:poke15
    63,32
690 poke1564,135:poke1565,180:poke1566
    ,160:poke1567,8:poke1568,52:poke15
    69,32
700 a=1586:gosub2530
710 a=1629:gosub2530
720 poke1666,1:poke1667,51:poke1668,32
    :poke1669,130:poke1670,179:poke167
    1,160
730 poke1672,3:poke1673,51:poke1674,32
    :poke1675,132:poke1676,179:poke167
    7,160
740 poke1678,5:poke1679,51:poke1680,32
    :poke1681,134:poke1682,179:poke168
    3,160
750 poke1684,7:poke1685,51:poke1686,32
    :poke1687,136:poke1688,179:poke168
    9,160
760 a=1709:gosub2530
770 a=1746:gosub2530
780 poke1786,129:poke1787,178:poke1788
    ,160:poke1789,2:poke1790,50:poke17
    91,32
790 poke1792,131:poke1793,178:poke1794
    ,160:poke1795,4:poke1796,50:poke17
    97,32
800 poke1798,133:poke1799,178:poke1800
    ,160:poke1801,6:poke1802,50:poke18
    03,32
810 poke1804,135:poke1805,178:poke1806
    ,160:poke1807,8:poke1808,50:poke18
    09,32
820 a=1826:gosub2530
830 a=1869:gosub2530
840 poke1906,1:poke1907,49:poke1908,32
    :poke1909,130:poke1910,177:poke191
    1,160
850 poke1912,3:poke1913,49:poke1914,32
```

```
     :poke1915,132:poke1916,177:poke191
     7,160
860  poke1918,5:poke1919,49:poke1920,32
     :poke1921,134:poke1922,177:poke192
     3,160
870  poke1924,7:poke1925,49:poke1926,32
     :poke1927,136:poke1928,177:poke192
     9,160
880  gosub1810
890  a=1949:gosub2530
900  v=53248
910  pokev+21,1
920  poke2040,13
930  pokev+39,0
940  pokev,40
950  pokev+1,50
960  x=40:y=50
970  print"[HOME][CRSR-DOWN]"tab(29)"ze
     t[SPACE]paard"
980  printtab(29)"begin[SPACE]pos."
990  printtab(29)"vuur!!!!!"
1000 print"[CTRL-8][7xCRSR-DOWN][SPACE]
     "tab(29)"toets[SPACE][CTRL-9]e[CTR
     L 0]"
1010 printtab(29)"voor[SPACE]eind"
1020 if dem=1  then goto  1880
1030 rem bewegen sprite
1040 if dem=1  then goto  1880
1050 geth$:if h$="e" then 2610
1060 s=peek(56321)
1070 if s=254 then 1130
1080 if s=253 then 1190
1090 if s=251 then 1250
1100 if s=247 then 1310
1110 if s=239 then 1370
1120 goto 1050
1130 rem omhoog
1140 if y<65 then 1050
1150 y=y-24
1160 pokev+1,y
1170 forz=0to100:next
1180 goto 1050
1190 rem omlaag
1200 if y>205 then 1050
1210 y=y+24
1220 pokev+1,y
1230 forz=0to100:next
1240 goto 1050
1250 rem naar links
1260 ifx<50then 1050
1270 x=x-24
1280 pokev,x
1290 forz=0to100:next
1300 goto1050
1310 rem naar rechts
1320 ifx>190then 1050
1330 x=x+24
1340 forz=0to100:next
1350 pokev,x
1360 goto 1050
1370 rem plaats bepaling
1380 r=int((y-50)/8+1):k=int((x-24)/8):
     n=1:rem aftelezen punt
1390 p=1024+r*40
1400 for z=k   to k+1
1410 s(n)=peek(p+z)
1420 n=n+1
1430 next
1440 s1=s(1):s2=s(2)
1450 z=0:zet=zet+1
1460 rem berekening
1470 if s1=1 or s1=129 then rij=1
1480 if s1=2 or s1=130then rij=2
1490 if s1=3 or s1=131 then rij=3
1500 if s1=4 or s1=132 then rij=4
1510 if s1=5 or s1=133 then rij=5
1520 if s1=6 or s1=134 then rij=6
1530 if s1=7 or s1=135 then rij=7
1540 if s1=8 or s1=136 then rij=8
1550 if s2=49 or s2=177 then cij=1
1560 if s2=50or s2=178 then cij=2
1570 if s2=51 or s2=179 then cij=3
1580 if s2=52 or s2=180then cij=4
1590 if s2=53 or s2=181 then cij=5
1600 if s2=54 or s2=182 then cij=6
1610 if s2=55 or s2=183 then cij=7
1620 if s2=56 or s2=184 then cij=8
1630 if zet>2 then goto 2020
1640 pz=rij:sz=cij:pl=rij:sl=cij
1650 pl$=chr$(pl+64):rl$=str$(sl):sl$=r
     ight$(rl$,1)
1660 m=m+1:c$(m)=pl$+sl$
1670 if zet>2 then 2290
1680 pa=rij:pb=cij
1690 pokep+k+2,90
1700 print"[HOME][CRSR-DOWN]"tab(29)"vo
     lgende--"
1710 printtab(29)"zet-------"
1720 printtab(29)"vuur!!!!"
1730 x1=x:y1=y:goto1030
1740 rem scherm rand
1750 print"[SHIFT-CLR]"
1760 for s=1025 to 1062:pokes,120:next:
     poke1024,105:poke1063,95
1770 fors=1064 to 1944 step 40:pokes,11
     7:next
1780 fors=1103to1983step40:pokes,118:ne
     xt
1790 for s=1985 to 2022:pokes,121:next:
     poke1984,77 :poke2023,78
1800 return
1810 rem hulpscherm
1820 fors=1053 to1062:pokes,119:next:po
     ke1052,105:poke1063,95
1830 fors=1092 to1972 step 40:pokes,117
     :next
1840 fors=1103 to 1983 step 40:pokes,11
     8:next
1850 fors=2013 to2022:pokes,121:next
1860 poke 2012,77:poke2023,78
1870 return
1880 rem demo
1890 oo=oo+1
1900 print"[HOME][CRSR-DOWN]"tab(29)"ze
     t[SPACE]paard"
1910 printtab(29)"op[SPACE]"  b$(oo)"--
     ---"
1920 printtab(29)"vuur!!!!"
1930 goto1050
1940 rem naam
1950 print"[4xCRSR-DOWN][4xCRSR-RIGHT][
     CTRL-8][CTRL-9]paard[SPACE]op[SPAC
     E]het[SPACE]schaakveld"
1960 print"[3xCRSR-DOWN][3xCRSR-RIGHT][
     SPACE][CTRL-9]jensje[SPACE]rienstr
     a"
```

```
1970 print"[3xCRSR-DOWN][3xCRSR-RIGHT]1989
1980 for x=0to 62:reada:poke832+x,a:nextx
1990 for o=1 to 64:reada$(o):next
2000 for o=1 to 64:readb$(o):next
2010 return
2020 rem cotrole zet
2030 print"[HOME][CRSR-DOWN]"tab(29)"ze
     t------"
2040 printtab(29)"controle--"
2050 printtab(29)"---------"
2060 if pz=rij+2 and sz=cij+1 then 2150
2070 if pz=rij+2 and sz=cij-1 then 2150
2080 if pz=rij-2 and sz=cij+1 then 2150
2090 if pz=rij-2 and sz=cij-1 then 2150
2100 if pz=rij+1 and sz=cij+2 then 2150
2110 if pz=rij+1 and sz=cij-2 then 2150
2120 if pz=rij-1 and sz=cij+2 then 2150
2130 if pz=rij-1 and sz=cij-2 then 2150
2140 goto2190
2150 print"[HOME][CRSR-DOWN]"tab(29)"vo
     lgende--"
2160 printtab(29)"zet-------"
2170 printtab(29)"vuur!!!!"
2180 goto1640
2190 print"[HOME][CRSR-DOWN]"tab(29)"ve
     rkeerde"
2200 printtab(29)"zet----"
2210 printtab(29)"--------"
2220 forz=0to1000:next
2230 print"[HOME][CRSR-DOWN]"tab(29)"te
     rug[SPACE]naar"
2240 printtab(29)"[CTRL-9]"c$(m)"[CTRL-
     0]volgende"
2250 printtab(29)"zet--vuur"
2260 x=x1:y=y1
2270 pokev,x:pokev+1,y
2280 goto 1030
2290 rem plaats controle
2300 print"[HOME][CRSR-DOWN]"tab(29)"pl
     aats----"
2310 printtab(29)"controle--"
2320 printtab(29)"---------"
2330 for aa=1 to m-1
2340 if c$(m)=c$(aa) then 2430
2350 print"[17xCRSR-DOWN][CTRL-9]"tab(2
     9) c$(aa)
2360 print"[18xCRSR-UP][2xCRSR-LEFT][CT
     RL 0]"
2370 next
2380 print"[HOME][CRSR-DOWN]"tab(29)"vo
     lgende--"
2390 printtab(29)"zet-------"
2400 printtab(29)"---------"
2410 goto1680
2420 :
2430 print"[HOME][CRSR-DOWN]"tab(29)c$(
     m)"[SPACE]is-----"
2440 printtab(29)"geweest---"
2450 printtab(29)"---------"
2460 for f=0to1000:next
2470 print"[HOME][CRSR-DOWN]"tab(29)"te
     rug[SPACE]naar"
2480 printtab(29)"[CTRL-9]"c$(m-1)"[CTR
     L 0]volgende"
2490 print tab(29)"zet--vuur"
2500 x=x1:y=y1:pz=pa:sz=pb
2510 pokev,x:pokev+1,y
2520 goto1030
2530 rem
2540 for c=0to 3
2550 for b=a to a+2
2560 poke b,160
2570 next
2580 a=a+6
2590 next c
2600 return
2610 rem eindcontrole
2620 x=0:y=0:pokev,x:pokev+1,y
2630 print"[SHIFT-CLR][5xCRSR-DOWN][CTR
     L 9]controle[SPACE]van[SPACE]de[SP
     ACE]door[SPACE]jou[SPACE]bezette[S
     PACE]vakken"
2640 print"[CTRL-9]de[SPACE]door[SPACE]
     jou[SPACE]bezette[SPACE]vakken[SPA
     CE]zijn"
2650 for l= 1 to 64
2660 for p= 1 to m
2670 if c$(p)=a$(l) then 2710
2680 nextp
2690 f$(w)=a$(l):w=w+1
2700 goto2730
2710 e$(rr)=c$(p):rr=rr+1
2720 print"[SPACE][CTRL-9]"e$(rr-1)"[SP
     ACE]";
2730 nextl
2740 print
2750 if rr=65 then print"[5xCRSR-DOWN][
     CTRL-9]het[SPACE]is[SPACE]je[SPACE
     ]gelukt" :goto2900
2760 print"[2xSPACE]niet[SPACE]gelukt"
2770 print"je[SPACE]hebt[SPACE]de[SPACE
     ]volgende[SPACE]plaatsen[SPACE]nie
     t[SPACE]bezet[SPACE]gehad"
2780 for u=1 to w
2790 print f$(u)"[SPACE]" ;
2800 next
2810 print"[4xCRSR-DOWN][4xCRSR-RIGHT]t
     oets[SPACE]voor[SPACE]vervolg"
2820 geth$:if h$="" then 2820
2830 print"[SHIFT-CLR][3xCRSR-DOWN][3xC
     RSR-RIGHT][CTRL-9]wil[SPACE]je[SPA
     CE]de[SPACE]volgorde[SPACE]van[SPA
     CE]jou[SPACE]zetten[SPACE]op[SPACE
     ]het[SPACE]scherm[SPACE](s)"
2840 print"[2xCRSR-DOWN][3xCRSR-RIGHT]w
     il[SPACE]je[SPACE]ze[SPACE]ook[SPA
     CE]afdrukken[SPACE]op[SPACE]de[SPA
     CE]printer[SPACE](p)"
2850 print"[4xCRSR-DOWN]toets[SPACE]voo
     r[SPACE]vervolg[SPACE](t)"
2860 get h$:if h$=""then 2860
2870 ifh$="s" then 3080
2880 if h$="p"then 3150
2890 print"[SHIFT-CLR]"
2900 print"[3xCRSR-DOWN]wil[SPACE]je[SP
     ACE]nog[SPACE]eens[SPACE]druk[SPAC
     E]dan[SPACE]op[CTRL-9]n[CTRL-0]"
2910 print"anders[SPACE]op[CTRL-9]s[CTR
     L-0]"
2920 geth$:if h$=""then 2920
2930 if h$="n" then 350
2940 if h$="s" then end
2950 ifh$<>"n" or h$<>"s" then 2920
2960 data0,128,0,0,192,0,1,224,0,1,252,
     0,3,62,0,7,63,0,7,255,0,15,255,128
2970 data 31,255,192,31,255,192,63,255,
```

```
     224,63,159,224,127,15,224
2980 data 14,31,240,12,31,240,0,63,248,
     0,63,248,0,127,252
2990 data0,255,252,1,255,255,1,255,255
3000 data a1,a2,a3,a4,a5,a6,a7,a8,b1,b2
     ,b3,b4,b5,b6,b7,b8
3010 data c1,c2,c3,c4,c5,c6,c7,c8,d1,d2
     ,d3,d4,d5,d6,d7,d8
3020 data e1,e2,e3,e4,e5,e6,e7,e8,f1,f2
     ,f3,f4,f5,f6,f7,f8
3030 data g1,g2,g3,g4,g5,g6,g7,g8,h1,h2
     ,h3,h4,h5,h6,h7,h8
3040 rem demo
3050 data c6,a7,c8,e7,d5,f4,h5,g3,h1,f2
     ,d3,e5,g4,h6,g8,f6,d7,b8,a6,b4,a2
3060 data c1,b3,a1,c2,a3,b1,c3,e2,g1,h3
     ,g5,h7,f8,g6,h8,f7,d8,b7,a5,c4,b2,d1
3070 data e3,f1,h2,f3,e1,g2,h4,f5,g7,e8
     ,d6,b5,d4,e6,c7,a8,b6,a4,c5,e4,d2
3080 rem afdrukken op scherm
3090 forss=1 to m
3100 printc$(ss)"[SPACE]";
3110 next
3120 print"[4xCRSR-DOWN][3xCRSR-RIGHT]t
     oets[SPACE]voor[SPACE]vervolg"
3130 get h$:if h$=""then 3230
3140 goto 2890
3150 rem afdrukken op printer en scherm
3160 open 4,4
3170 for ss=1 to m
3180 rem  print c$(ss)"[SPACE]";
3190 print#4,c$(ss)"[SPACE]"
3200 next
3210 close 4
3220 print"[4xCRSR-DOWN][3xCRSR-RIGHT]t
     oets[SPACE]voor[SPACE]vervolg"
3230 get h$:if h$=""then 3230
3240 goto2890
3250 rem****************************
3260 rem**                        **
3270 rem**     J.D. Rienstra     **
3280 rem**         SNEEK          **
3290 rem**       (C)  1989        **
3300 rem**                        **
3310 rem****************************
```

** EINDE LISTING paard

Checksum paard

REGEL 10	95	REGEL 250	89	REGEL 490	92
REGEL 20	65	REGEL 260	219	REGEL 500	60
REGEL 30	207	REGEL 270	209	REGEL 510	64
REGEL 40	114	REGEL 280	51	REGEL 520	82
REGEL 50	96	REGEL 290	48	REGEL 530	83
REGEL 60	239	REGEL 300	194	REGEL 540	59
REGEL 70	97	REGEL 310	55	REGEL 550	54
REGEL 80	91	REGEL 320	112	REGEL 560	58
REGEL 90	238	REGEL 330	240	REGEL 570	80
REGEL 100	89	REGEL 340	253	REGEL 580	82
REGEL 110	235	REGEL 350	190	REGEL 590	89
REGEL 120	238	REGEL 360	145	REGEL 600	66
REGEL 130	79	REGEL 370	4	REGEL 610	70
REGEL 140	28	REGEL 380	110	REGEL 620	74
REGEL 150	223	REGEL 390	37	REGEL 630	96
REGEL 160	114	REGEL 400	166	REGEL 640	88
REGEL 170	89	REGEL 410	77	REGEL 650	80
REGEL 180	83	REGEL 420	81	REGEL 660	91
REGEL 190	180	REGEL 430	76	REGEL 670	86
REGEL 200	65	REGEL 440	80	REGEL 680	90
REGEL 210	75	REGEL 450	102	REGEL 690	112
REGEL 220	199	REGEL 460	76	REGEL 700	88
REGEL 230	136	REGEL 470	83	REGEL 710	86
REGEL 240	113	REGEL 480	88	REGEL 720	107

REGEL 730	111	REGEL 1620	24	REGEL 2510	248
REGEL 740	115	REGEL 1630	85	REGEL 2520	77
REGEL 750	137	REGEL 1640	126	REGEL 2530	143
REGEL 760	85	REGEL 1650	132	REGEL 2540	125
REGEL 770	86	REGEL 1660	69	REGEL 2550	119
REGEL 780	132	REGEL 1670	213	REGEL 2560	156
REGEL 790	127	REGEL 1680	124	REGEL 2570	130
REGEL 800	95	REGEL 1690	77	REGEL 2580	20
REGEL 810	99	REGEL 1700	42	REGEL 2590	197
REGEL 820	85	REGEL 1710	66	REGEL 2600	142
REGEL 830	92	REGEL 1720	234	REGEL 2610	21
REGEL 840	94	REGEL 1730	233	REGEL 2620	225
REGEL 850	98	REGEL 1740	118	REGEL 2630	160
REGEL 860	102	REGEL 1750	112	REGEL 2640	168
REGEL 870	124	REGEL 1760	233	REGEL 2650	190
REGEL 880	87	REGEL 1770	121	REGEL 2660	165
REGEL 890	91	REGEL 1780	119	REGEL 2670	184
REGEL 900	14	REGEL 1790	219	REGEL 2680	210
REGEL 910	87	REGEL 1800	142	REGEL 2690	59
REGEL 920	237	REGEL 1810	138	REGEL 2700	85
REGEL 930	95	REGEL 1820	243	REGEL 2710	39
REGEL 940	125	REGEL 1830	123	REGEL 2720	168
REGEL 950	89	REGEL 1840	119	REGEL 2730	206
REGEL 960	24	REGEL 1850	86	REGEL 2740	153
REGEL 970	215	REGEL 1860	41	REGEL 2750	157
REGEL 980	153	REGEL 1870	142	REGEL 2760	217
REGEL 990	11	REGEL 1880	180	REGEL 2770	199
REGEL 1000	229	REGEL 1890	201	REGEL 2780	180
REGEL 1010	122	REGEL 1900	215	REGEL 2790	40
REGEL 1020	69	REGEL 1910	45	REGEL 2800	130
REGEL 1030	99	REGEL 1920	234	REGEL 2810	143
REGEL 1040	69	REGEL 1930	79	REGEL 2820	167
REGEL 1050	233	REGEL 1940	172	REGEL 2830	5
REGEL 1060	25	REGEL 1950	3	REGEL 2840	225
REGEL 1070	151	REGEL 1960	160	REGEL 2850	192
REGEL 1080	156	REGEL 1970	66	REGEL 2860	171
REGEL 1090	151	REGEL 1980	186	REGEL 2870	178
REGEL 1100	153	REGEL 1990	67	REGEL 2880	173
REGEL 1110	160	REGEL 2000	68	REGEL 2890	112
REGEL 1120	79	REGEL 2010	142	REGEL 2900	52
REGEL 1130	88	REGEL 2020	154	REGEL 2910	48
REGEL 1140	111	REGEL 2030	125	REGEL 2920	168
REGEL 1150	117	REGEL 2040	212	REGEL 2930	122
REGEL 1160	77	REGEL 2050	169	REGEL 2940	103
REGEL 1170	174	REGEL 2060	214	REGEL 2950	120
REGEL 1180	79	REGEL 2070	215	REGEL 2960	156
REGEL 1190	64	REGEL 2080	215	REGEL 2970	248
REGEL 1200	153	REGEL 2090	216	REGEL 2980	120
REGEL 1210	116	REGEL 2100	214	REGEL 2990	26
REGEL 1220	77	REGEL 2110	215	REGEL 3000	119
REGEL 1230	174	REGEL 2120	215	REGEL 3010	151
REGEL 1240	79	REGEL 2130	216	REGEL 3020	183
REGEL 1250	50	REGEL 2140	85	REGEL 3030	215
REGEL 1260	104	REGEL 2150	42	REGEL 3040	180
REGEL 1270	115	REGEL 2160	66	REGEL 3050	240
REGEL 1280	113	REGEL 2170	234	REGEL 3060	116
REGEL 1290	174	REGEL 2180	84	REGEL 3070	225
REGEL 1300	79	REGEL 2190	25	REGEL 3080	139
REGEL 1310	122	REGEL 2200	187	REGEL 3090	251
REGEL 1320	155	REGEL 2210	124	REGEL 3100	118
REGEL 1330	114	REGEL 2220	222	REGEL 3110	130
REGEL 1340	174	REGEL 2230	37	REGEL 3120	114
REGEL 1350	113	REGEL 2240	85	REGEL 3130	163
REGEL 1360	79	REGEL 2250	179	REGEL 3140	92
REGEL 1370	150	REGEL 2260	98	REGEL 3150	66
REGEL 1380	121	REGEL 2270	248	REGEL 3160	51
REGEL 1390	213	REGEL 2280	77	REGEL 3170	251
REGEL 1400	162	REGEL 2290	186	REGEL 3180	249
REGEL 1410	11	REGEL 2300	245	REGEL 3190	154
REGEL 1420	41	REGEL 2310	212	REGEL 3200	130
REGEL 1430	130	REGEL 2320	169	REGEL 3210	212
REGEL 1440	82	REGEL 2330	179	REGEL 3220	114
REGEL 1450	233	REGEL 2340	236	REGEL 3230	163
REGEL 1460	105	REGEL 2350	129	REGEL 3240	92
REGEL 1470	227	REGEL 2360	219	REGEL 3250	39
REGEL 1480	221	REGEL 2370	130	REGEL 3260	137
REGEL 1490	224	REGEL 2380	42	REGEL 3270	252
REGEL 1500	227	REGEL 2390	66	REGEL 3280	173
REGEL 1510	230	REGEL 2400	169	REGEL 3290	150
REGEL 1520	233	REGEL 2410	88	REGEL 3300	97
REGEL 1530	236	REGEL 2420	58	REGEL 3310	39
REGEL 1540	239	REGEL 2430	254		
REGEL 1550	21	REGEL 2440	175		
REGEL 1560	15	REGEL 2450	169		
REGEL 1570	18	REGEL 2460	202		
REGEL 1580	12	REGEL 2470	37		
REGEL 1590	15	REGEL 2480	49		
REGEL 1600	18	REGEL 2490	179		
REGEL 1610	21	REGEL 2500	180		

PRINT OUT C-128 met Kaartspel 4x13

Kaartspel 4x13

Dit spel, gemaakt door Dirk Vereyck uit België, is een kaartspel waarbij men de 13 kaarten van een bepaald soort op 1 rij moet trachten te leggen. Dit moet dan wel volgens een bepaalde methode gebeuren. Eerst worden de vier AZEN op het eerste lege vakje geplaatst. Daarna zijn in de lege vakjes die zijn onstaan een kaart leggen. Een kaart wordt alleen toegestaan als deze 1 hoger is in waarde dan de voorgaande kaart en van het zelfde soort is. Op deze maniet moeten de vier rijen in volgorde worden neergelegt. Loopt men halve wege vast dan kunnen de kaarten worden geschud. Men speelt me de joystick in poort twee, gestopt kan er eventueel worden door het indrukken van de RUN/STOP toets.

```
10  rem"[F1][SHIFT-CLR]list
20  rem "[2xSPACE]Kaartspel[SPACE]13[S
    PACE]x[SPACE]4[2xSPACE]"
30  rem "Vereyck[SPACE]Dirk[SPACE]-[SP
    ACE]Beveren-Yzer[SPACE](Belgie)"
40  rem
50  scnclr:dimk$(52),h$(52),ry$(4),ks$
    (14,4),r(4):be=1:color0,8:color4,8
    :sprite1,1,8:
60  rem kaarten array vullen
70  fori=1to52:readk$(i):next
80  data "1A","2A","3A","4A","5A","6A"
    ,"7A","8A","9A","tA","vA","dA","rA
    "
90  data "1S","2S","3S","4S","5S","6S"
    ,"7S","8S","9S","tS","vS","dS","rS
    "
100 data "1X","2X","3X","4X","5X","6X"
    ,"7X","8X","9X","tX","vX","dX","rX
    "
110 data "1Z","2Z","3Z","4Z","5Z","6Z"
    ,"7Z","8Z","9Z","tZ","vZ","dZ","rZ
    "
120 gosub2500
130 rem hulp array invullen
140 fori=1to52:h$(i)=k$(i):nexti
150 rem kaarten array 13 x 4 maken (rn
    d functie)
160 fori=1to4
170 :do
180 :iflen(ry$(i))=26thenexit
190 :x=int(rnd(0)*52)+1:ifh$(x)=""then
    190
200 :ry$(i)=ry$(i)+h$(x):h$(x)=""
210 :loop
220 nexti
230 fori=1to4:forj=0to12:ks$(j+1,i)=mi
    d$(ry$(i),j*2+1,2):nextj,i
240 rem sprite - pijltje
250 scnclr:graphic1,0:color4,8:color0,
    8:color1,8
260 draw1,0,0to0,7to7,0:paint1,1,1:dra
    w1,2,0to15,12to13,14to0,2:paint1,9
    ,9
270 shapez$,0,0,23,21
280 sprsavez$,1
290 graphic0,1
300 rem *** schermopbouw ***
310 graphic1,1:color4,8:color0,8
320 color1,5:box1,91,5,229,17:char1,12
    ,1,"kaartspel[SPACE]4[SPACE]x[SPAC
    E]13"
330 forj=0to3:box1,6,29+j*32,25,40+j*3
    2:nextj
340 forj=0to3:fori=0to12:box1,6+24*i,4
    5+j*32,25+24*i,56+j*32:nexti,j
350 char1,27,22,"husselen":char1,27,23
    ,"nieuw[SPACE]spel":char1,27,24,"s
    toppen[SPACE]spel"
360 fori=0to2:box1,206,176+8*i,213,182
    +8*i:nexti
370 forj=0to3:fori=0to12:gosub390:char
    1,1+(i*3),6+j*4,ks$(i+1,j+1):nexti
    ,j
380 goto420
390 ifright$(ks$(i+1,j+1),1)="A"orrigh
    t$(ks$(i+1,j+1),1)="X"thencolor1,1
400 ifright$(ks$(i+1,j+1),1)="S"orrigh
    t$(ks$(i+1,j+1),1)="Z"thencolor1,3
410 return
420 rem ** start spel **
430 color1,5:char1,2,23,"beurt"+str$(b
    e):char1,2,24,"[21xSPACE]"
440 x=190:y=70:t=1
450 sprite1,1,6:movespr1,x,y
460 ifjoy(2)=1theny=y-3
470 ifjoy(2)=5theny=y+3
480 ifjoy(2)=3thenx=x+3
490 ifjoy(2)=7thenx=x-3
500 ifjoy(2)=8thenx=x-3:y=y-3
510 ifjoy(2)=2thenx=x+3:y=y-3
520 ifjoy(2)=4thenx=x+3:y=y+3
530 ifjoy(2)=6thenx=x-3:y=y+3
540 ifjoy(2)=128then560
550 goto450
560 x1=rsppos(1,0)
570 y1=rsppos(1,1)
580 ifx1>31andx1<49andy1>79andy1<91the
    nr=1:k=0:goto1190
590 ifx1>31andx1<49andy1>95andy1<106th
    enr=1:k=1:goto1190
600 ifx1>55andx1<73andy1>95andy1<106th
    enr=1:k=2:goto1190
610 ifx1>79andx1<97andy1>95andy1<106th
    enr=1:k=3:goto1190
620 ifx1>102andx1<121andy1>95andy1<106
    thenr=1:k=4:goto1190
630 ifx1>127andx1<145andy1>95andy1<106
    thenr=1:k=5:goto1190
640 ifx1>151andx1<169andy1>95andy1<106
    thenr=1:k=6:goto1190
650 ifx1>175andx1<193andy1>95andy1<106
    thenr=1:k=7:goto1190
660 ifx1>199andx1<217andy1>95andy1<106
    thenr=1:k=8:goto1190
670 ifx1>223andx1<241andy1>95andy1<106
    thenr=1:k=9:goto1190
680 ifx1>247andx1<265andy1>95andy1<106
    thenr=1:k=10:goto1190
690 ifx1>271andx1<289andy1>95andy1<106
    thenr=1:k=11:goto1190
700 ifx1>295andx1<313andy1>95andy1<106
    thenr=1:k=12:goto1190
710 ifx1>319andx1<337andy1>95andy1<106
    thenr=1:k=13:goto1190
```

```
720  ifx1>31andx1<49andy1>112andy1<122t
     henr=2:k=0:goto1190
730  ifx1>31andx1<49andy1>128andy1<139t
     henr=2:k=1:goto1190
740  ifx1>55andx1<73andy1>128andy1<139t
     henr=2:k=2:goto1190
750  ifx1>79andx1<97andy1>128andy1<139t
     henr=2:k=3:goto1190
760  ifx1>102andx1<121andy1>128andy1<13
     9thenr=2:k=4:goto1190
770  ifx1>127andx1<145andy1>128andy1<13
     9thenr=2:k=5:goto1190
780  ifx1>151andx1<169andy1>128andy1<13
     9thenr=2:k=6:goto1190
790  ifx1>175andx1<193andy1>128andy1<13
     9thenr=2:k=7:goto1190
800  ifx1>199andx1<217andy1>128andy1<13
     9thenr=2:k=8:goto1190
810  ifx1>223andx1<241andy1>128andy1<13
     9thenr=2:k=9:goto1190
820  ifx1>247andx1<265andy1>128andy1<13
     9thenr=2:k=10:goto1190
830  ifx1>271andx1<289andy1>128andy1<13
     9thenr=2:k=11:goto1190
840  ifx1>295andx1<313andy1>128andy1<13
     9thenr=2:k=12:goto1190
850  ifx1>319andx1<337andy1>128andy1<13
     9thenr=2:k=13:goto1190
860  ifx1>31andx1<49andy1>143andy1<154t
     henr=3:k=0:goto1190
870  ifx1>31andx1<49andy1>160andy1<171t
     henr=3:k=1:goto1190
880  ifx1>55andx1<73andy1>160andy1<171t
     henr=3:k=2:goto1190
890  ifx1>79andx1<97andy1>160andy1<171t
     henr=3:k=3:goto1190
900  ifx1>102andx1<121andy1>160andy1<17
     1thenr=3:k=4:goto1190
910  ifx1>127andx1<145andy1>160andy1<17
     1thenr=3:k=5:goto1190
920  ifx1>151andx1<169andy1>160andy1<17
     1thenr=3:k=6:goto1190
930  ifx1>175andx1<193andy1>160andy1<17
     1thenr=3:k=7:goto1190
940  ifx1>199andx1<217andy1>160andy1<17
     1thenr=3:k=8:goto1190
950  ifx1>223andx1<241andy1>160andy1<17
     1thenr=3:k=9:goto1190
960  ifx1>247andx1<265andy1>160andy1<17
     1thenr=3:k=10:goto1190
970  ifx1>271andx1<289andy1>160andy1<17
     1thenr=3:k=11:goto1190
980  ifx1>295andx1<313andy1>160andy1<17
     1thenr=3:k=12:goto1190
990  ifx1>319andx1<337andy1>160andy1<17
     1thenr=3:k=13:goto1190
1000 ifx1>31andx1<49andy1>176andy1<187t
     henr=4:k=0:goto1190
1010 ifx1>31andx1<49andy1>191andy1<202t
     henr=4:k=1:goto1190
1020 ifx1>55andx1<73andy1>191andy1<202t
     henr=4:k=2:goto1190
1030 ifx1>79andx1<97andy1>191andy1<202t
     henr=4:k=3:goto1190
1040 ifx1>102andx1<121andy1>191andy1<20
     2thenr=4:k=4:goto1190
1050 ifx1>127andx1<145andy1>191andy1<20
     2thenr=4:k=5:goto1190
1060 ifx1>151andx1<169andy1>191andy1<20
     2thenr=4:k=6:goto1190
1070 ifx1>175andx1<193andy1>191andy1<20
     2thenr=4:k=7:goto1190
1080 ifx1>199andx1<217andy1>191andy1<20
     2thenr=4:k=8:goto1190
1090 ifx1>223andx1<241andy1>191andy1<20
     2thenr=4:k=9:goto1190
1100 ifx1>247andx1<265andy1>191andy1<20
     2thenr=4:k=10:goto1190
1110 ifx1>271andx1<289andy1>191andy1<20
     2thenr=4:k=11:goto1190
1120 ifx1>295andx1<313andy1>191andy1<20
     2thenr=4:k=12:goto1190
1130 ifx1>319andx1<337andy1>191andy1<20
     2thenr=4:k=13:goto1190
1140 ifx1>231andx1<238andy1>226andy1<23
     2then1590
1150 ifx1>231andx1<238andy1>234andy1<24
     1thengraphic1,0:run
1160 ifx1>231andx1<238andy1>243andy1<24
     9then2340
1170 goto450
1180 rem karakter in de dim-string verp
     laatsen
1190 ift=1then1200:else1230
1200 r1$=ks$(k,r):z1=r:z2=k
1210 gosub1550:char1,2,21,r1$:color1,5:
     char1,5,21,"naar[SPACE]??"
1220 t=2:goto450
1230 z3=r:z4=k
1240 ifz4=0then1290
1250 goto1320:rem controle op juistheid
1260 ks$(z4,z3)=r1$:char1,1+(z2-1)*3,6+
     (z1-1)*4,"[2xSPACE]"
1270 gosub1550:char1,1+(z4-1)*3,6+(z3-1
     )*4,r1$
1280 ks$(z2,z1)="":t=1:char1,2,21,"[11x
     SPACE]":goto450
1290 ks$(z4,z3)=r1$:char1,1+(z2-1)*3,6+
     (z1-1)*4,"[2xSPACE]"
1300 gosub1550:char1,1,4+(z3-1)*4,r1$
1310 ks$(z2,z1)="":t=1:char1,2,21,"[11x
     SPACE]":goto450
1320 rem controle van de zet
1330 ifks$(z4,z3)<>""then1490
1340 r2$=ks$(z4-1,z3):t$=right$(r2$,1)
1350 ift$<>right$(r1$,1)then1490
1360 t1$=left$(r2$,1):t2$=left$(r1$,1):
     t3$=t1$+t2$
1370 ift3$="12"then1260
1380 ift3$="23"then1260
1390 ift3$="34"then1260
1400 ift3$="45"then1260
1410 ift3$="56"then1260
1420 ift3$="67"then1260
1430 ift3$="78"then1260
1440 ift3$="89"then1260
1450 ift3$="9t"then1260
1460 ift3$="tv"then1260
1470 ift3$="vd"then1260
1480 ift3$="dr"then1260
1490 rem foutmelding + geluid
1500 color1,5:char1,2,21,"fout[SPACE]!!
     [SPACE]gaat[SPACE]niet[SPACE]!!":s
     ound1,49152,60,2,32768,3000,1
1510 fori=1to750:next:char1,2,21,"[22xS
     PACE]"
```

```
1520 t=1:goto450
1530 end
1540 rem gosub kleurbepaling
1550 ifright$(r1$,1)="A"orright$(r1$,1)
     ="X"thencolor1,1
1560 ifright$(r1$,1)="S"orright$(r1$,1)
     ="Z"thencolor1,3
1570 return
1580 ify=0theny=1:goto420
1590 gosub2290
1600 rem controle op volledige rijen
1610 fori=1to4:e$="":forj=0to13
1620 c1$=right$(ks$(0,i),1)
1630 c$="1"+c1$+"2"+c1$+"3"+c1$+"4"+c1$
     +"5"+c1$+"6"+c1$+"7"+c1$+"8"+c1$+"
     9"+c1$+"t"+c1$+"v"+c1$+"d"+c1$+"r"
     +c1$
1640 e$=e$+ks$(j,i)
1650 nextj
1660 ife$=c$thenok(i)=1:elseok(i)=0
1670 nexti
1680 rem controle kaart die niet meer v
     olgt
1690 forj=1to4
1700 ifok(j)=1then1900
1710 fori=0to13
1720 s1$=ks$(i,j):s2$=ks$(i+1,j)
1730 ifright$(s1$,1)<>right$(s2$,1)then
     1870
1740 u$=left$(s1$,1)+left$(s2$,1)
1750 ifu$="12"then1890
1760 ifu$="23"then1890
1770 ifu$="34"then1890
1780 ifu$="45"then1890
1790 ifu$="56"then1890
1800 ifu$="67"then1890
1810 ifu$="78"then1890
1820 ifu$="89"then1890
1830 ifu$="9t"then1890
1840 ifu$="tv"then1890
1850 ifu$="vd"then1890
1860 ifu$="dr"then1890
1870 r(j)=i+1
1880 goto1900
1890 nexti
1900 nextj
1910 rem bepaling aantal restkaarten
1920 k=0:fori=1to4
1930 ifok(i)=1then1950
1940 k=k+(13-(r(i)-1))
1950 nexti
1960 rem restkaarten in h$
1970 l=1
1980 forj=1to4
1990 ifok(j)=1then2030
2000 fori=r(j)to13
2010 h$(l)=ks$(i,j):ks$(i,j)="":l=l+1
2020 nexti
2030 nextj
2040 rem husselen
2050 fori=1to4
2060 ifok(i)=1then2120
2070 ifr(i)=13then2120
2080 forj=r(i)+1to13
2090 x=int(rnd(0)*k)+1:ifh$(x)=""then20
     90
2100 ks$(j,i)=h$(x):h$(x)=""
2110 nextj
2120 nexti
2130 rem restkaarten weg van scherm
2140 fori=1to4
2150 ifok(i)=1then2190
2160 forj=r(i)to13
2170 char1,1+(j-1)*3,6+(i-1)*4,"[2xSPAC
     E]"
2180 nextj
2190 nexti
2200 rem de gehusselde kaarten op het s
     cherm
2210 fori=1to4
2220 ifok(i)=1then2270
2230 ifr(i)=13then2270
2240 forj=r(i)to13
2250 r1$=ks$(j,i):gosub1550:char1,1+(j-
     1)*3,6+(i-1)*4,r1$
2260 nextj
2270 nexti
2280 goto420
2290 y=1:rem beurt=beurt+1
2300 y=1:rem beurt=beurt+1
2310 ifbe=3thencolor1,5:char1,2,24,"jam
     mer,niet[SPACE]gelukt[SPACE]!!":y=
     0
2320 be=be+1
2330 return
2340 rem stoppen spel
2350 rem controle op volledige rijen
2360 fori=1to4:e$="":forj=0to13
2370 c1$=right$(ks$(0,i),1)
2380 c$="1"+c1$+"2"+c1$+"3"+c1$+"4"+c1$
     +"5"+c1$+"6"+c1$+"7"+c1$+"8"+c1$+"
     9"+c1$+"t"+c1$+"v"+c1$+"d"+c1$+"r"
     +c1$
2390 e$=e$+ks$(j,i)
2400 nextj
2410 ife$=c$thenok(i)=1:elseok(i)=0
2420 nexti
2430 fori=1to4:o=o+ok(i):next
2440 ifo<>4thench$="niet[SPACE]volledig
     [SPACE]!![5xSPACE]":goto2470
2450 ifbe<=3thench$="proficiat[SPACE]!!
     [9xSPACE]":goto2470
2460 ch$="ok.[SPACE]goed[SPACE]!![12xSP
     ACE]"
2470 color1,5:char1,2,24,ch$
2480 x=235:y=237
2490 goto450
2500 rem inleidingsscherm
2510 graphic1,1:color4,8:color0,8
2520 color1,5:box1,91,5,229,17:char1,12
     ,1,"kaartspel[SPACE]4[SPACE]x[SPAC
     E]13"
2530 forj=0to3:box1,6,29+j*32,25,40+j*3
     2:nextj
2540 forj=0to3:fori=0to12:box1,6+24*i,4
     5+j*32,25+24*i,56+j*32:nexti,j
2550 forj=0to3:x=13*j:fori=0to12
2560 ifi=0thenr1$=k$(i+1+x):gosub1550:c
     har1,1+(i*3),4+(j*4),r1$:goto2580
2570 r1$=k$(i+1+x):gosub1550:char1,1+((
     i-1)*3),6+(j*4),r1$
2580 nexti
2590 nextj
2600 color1,5:char1,0,20,"bedoeling[SPA
     CE]:[SPACE]volledige[SPACE]rijen[S
     PACE]maken."
```

```
2610 char1,0,21,"werkwijze[SPACE]:[SPAC
     E]bv.[SPACE]na":color1,1:char1,19,
     21,"2A":color1,5:char1,23,21,"moet
     "
2620 color1,1:char1,28,21,"3A":color1,5
     :char1,31,21,"komen[SPACE]in":char
     1,16,22,"het[SPACE]leeg[SPACE]vakj
     e[SPACE]erachter."
2630 char1,0,23,"opm.[SPACE]:[SPACE]t=1
     0[SPACE]-[SPACE]v=boer[SPACE]-[SPA
     CE]d=dame[SPACE]-[SPACE]r=heer"
2640 color1,6:char1,0,24,"veel[SPACE]ge
     luk[SPACE]en[SPACE]natuurlijk[SPAC
     E]goede[SPACE]kaarten[SPACE]!"
2650 fori=1to2000:next
2660 return
```

** EINDE LISTING kaart **

Checksum kaartspel 4 x 13

REGEL 10	146	REGEL 720	82	REGEL 1430	11	REGEL 2140	133
REGEL 20	234	REGEL 730	98	REGEL 1440	13	REGEL 2150	21
REGEL 30	58	REGEL 740	102	REGEL 1450	41	REGEL 2160	113
REGEL 40	143	REGEL 750	115	REGEL 1460	70	REGEL 2170	64
REGEL 50	35	REGEL 760	187	REGEL 1470	54	REGEL 2180	204
REGEL 60	234	REGEL 770	201	REGEL 1480	50	REGEL 2190	203
REGEL 70	62	REGEL 780	205	REGEL 1490	178	REGEL 2200	73
REGEL 80	241	REGEL 790	209	REGEL 1500	24	REGEL 2210	133
REGEL 90	219	REGEL 800	213	REGEL 1510	81	REGEL 2220	20
REGEL 100	28	REGEL 810	199	REGEL 1520	147	REGEL 2230	255
REGEL 110	54	REGEL 820	251	REGEL 1530	128	REGEL 2240	113
REGEL 120	84	REGEL 830	255	REGEL 1540	212	REGEL 2250	154
REGEL 130	180	REGEL 840	250	REGEL 1550	24	REGEL 2260	204
REGEL 140	184	REGEL 850	254	REGEL 1560	46	REGEL 2270	203
REGEL 150	179	REGEL 860	92	REGEL 1570	142	REGEL 2280	31
REGEL 160	133	REGEL 870	91	REGEL 1580	2	REGEL 2290	162
REGEL 170	37	REGEL 880	95	REGEL 1590	90	REGEL 2300	162
REGEL 180	240	REGEL 890	108	REGEL 1600	167	REGEL 2310	93
REGEL 190	133	REGEL 900	180	REGEL 1610	13	REGEL 2320	155
REGEL 200	194	REGEL 910	194	REGEL 1620	121	REGEL 2330	142
REGEL 210	38	REGEL 920	198	REGEL 1630	252	REGEL 2340	236
REGEL 220	203	REGEL 930	202	REGEL 1640	0	REGEL 2350	167
REGEL 230	152	REGEL 940	206	REGEL 1650	204	REGEL 2360	13
REGEL 240	165	REGEL 950	192	REGEL 1660	240	REGEL 2370	121
REGEL 250	177	REGEL 960	244	REGEL 1670	203	REGEL 2380	252
REGEL 260	194	REGEL 970	248	REGEL 1680	31	REGEL 2390	0
REGEL 270	58	REGEL 980	243	REGEL 1690	134	REGEL 2400	204
REGEL 280	239	REGEL 990	247	REGEL 1700	20	REGEL 2410	240
REGEL 290	107	REGEL 1000	105	REGEL 1710	180	REGEL 2420	203
REGEL 300	41	REGEL 1010	91	REGEL 1720	110	REGEL 2430	169
REGEL 310	218	REGEL 1020	95	REGEL 1730	165	REGEL 2440	22
REGEL 320	87	REGEL 1030	108	REGEL 1740	18	REGEL 2450	105
REGEL 330	252	REGEL 1040	180	REGEL 1750	214	REGEL 2460	206
REGEL 340	109	REGEL 1050	194	REGEL 1760	216	REGEL 2470	143
REGEL 350	233	REGEL 1060	198	REGEL 1770	218	REGEL 2480	133
REGEL 360	152	REGEL 1070	202	REGEL 1780	220	REGEL 2490	34
REGEL 370	4	REGEL 1080	206	REGEL 1790	222	REGEL 2500	55
REGEL 380	31	REGEL 1090	192	REGEL 1800	224	REGEL 2510	218
REGEL 390	218	REGEL 1100	244	REGEL 1810	226	REGEL 2520	87
REGEL 400	240	REGEL 1110	248	REGEL 1820	228	REGEL 2530	252
REGEL 410	142	REGEL 1120	243	REGEL 1830	0	REGEL 2540	109
REGEL 420	249	REGEL 1130	247	REGEL 1840	29	REGEL 2550	15
REGEL 430	150	REGEL 1140	96	REGEL 1850	13	REGEL 2560	227
REGEL 440	193	REGEL 1150	191	REGEL 1860	9	REGEL 2570	35
REGEL 450	109	REGEL 1160	97	REGEL 1870	195	REGEL 2580	203
REGEL 460	169	REGEL 1170	34	REGEL 1880	83	REGEL 2590	204
REGEL 470	172	REGEL 1180	39	REGEL 1890	203	REGEL 2600	142
REGEL 480	168	REGEL 1190	1	REGEL 1900	204	REGEL 2610	2
REGEL 490	173	REGEL 1200	193	REGEL 1910	198	REGEL 2620	176
REGEL 500	42	REGEL 1210	97	REGEL 1920	236	REGEL 2630	65
REGEL 510	35	REGEL 1220	148	REGEL 1930	24	REGEL 2640	238
REGEL 520	36	REGEL 1230	86	REGEL 1940	107	REGEL 2650	207
REGEL 530	39	REGEL 1240	110	REGEL 1950	203	REGEL 2660	14
REGEL 540	108	REGEL 1250	198	REGEL 1960	214		
REGEL 550	34	REGEL 1260	177	REGEL 1970	47		
REGEL 560	236	REGEL 1270	189	REGEL 1980	134		
REGEL 570	238	REGEL 1280	193	REGEL 1990	15		
REGEL 580	2	REGEL 1290	177	REGEL 2000	113		
REGEL 590	46	REGEL 1300	119	REGEL 2010	238		
REGEL 600	50	REGEL 1310	193	REGEL 2020	203		
REGEL 610	63	REGEL 1320	86	REGEL 2030	204		
REGEL 620	135	REGEL 1330	2	REGEL 2040	246		
REGEL 630	149	REGEL 1340	19	REGEL 2050	133		
REGEL 640	153	REGEL 1350	250	REGEL 2060	14		
REGEL 650	157	REGEL 1360	192	REGEL 2070	249		
REGEL 660	161	REGEL 1370	255	REGEL 2080	76		
REGEL 670	147	REGEL 1380	1	REGEL 2090	96		
REGEL 680	199	REGEL 1390	3	REGEL 2100	222		
REGEL 690	203	REGEL 1400	5	REGEL 2110	204		
REGEL 700	198	REGEL 1410	7	REGEL 2120	203		
REGEL 710	202	REGEL 1420	9	REGEL 2130	93		

PRINT OUT Amiga met o.a. Alert

Alert

De behandeling van Alert

Guru! Guru! Ik verlang naar Kickstart v1.4 waar deze dingen nu eindelijk afgeschaft worden (die dingen met grote rode knipperende balken er om). De Alert is er omdat deze minder geheugen kost dan bijvoorbeeld een AutoRequest zodat deze ook aangeroepen kan worden als er bij het reserveren van geheugen bleek dat er niet genoeg was en dat er dan toch nog een boodschap moet verschijven (zoals Out of Memory...). Met deze routine kun je dus zelf van die leuke dingen maken. Verdere informatie staat in het programmakommentaar..

```
DECLARE FUNCTION AutoRequest&  LIBRARY
DECLARE FUNCTION AllocMem&     LIBRARY
DECLARE FUNCTION DisplayAlert& LIBRARY

' CInfo 24 Oktober 1989
' Modules Request en Alert
'========================
' AmigaBASIC ondersteuning van de
functies AutoRequest en DisplayAlert
' Voor eventuele problemen : R.Sloot /
01659-3686
' Haal A.U.B. voor uw eigen versie het
kommentaar eruit zodat
' het geheel (een beetje) sneller
loopt.

libpath$="VD0:"
LIBRARY libpath$+"graphics.library"
LIBRARY libpath$+"exec.library"
LIBRARY libpath$+"intuition.library"

'(*AC) voor een regel houdt in dat bij
gebruik van de AC-BASIC
'compiler deze regel in het programma
moet (' teken weghalen)
'(*AMG) aan het eind van een regel
houdt in dat bij gebruik van
' de AC-BASIC compiler deze regel uit
het programma moet

'(*AC) DIM ACString$(500/20)
' opmerking: buffergrootte (in
Request)/grootte van intuitext
' struktuur (even)

RESTORE Alert1Data
Alert 90&
RESTORE Alert2Data
Alert 60&
LOCATE 18
IF response&=0 THEN
  PRINT "Guru Guru 00000003 Help!"
ELSE
  PRINT "Guru Guru 81000009 Dit is een
overduidelijk programmeer foutje!"
END IF

win&=WINDOW(7)
RESTORE Voorbeeld1Data
Request win&,260&,70&

IF errnr%=0 THEN
  LOCATE 20
  IF response&=1 THEN
    PRINT "Er is gedrukt op het linkse
gadget (POSITIEF)."
  ELSE
    PRINT "U heeft op het rechtse
(NEGATIEF) gadget gedrukt."
  END IF
ELSE
  PRINT "FoutCode ";STR$(errnr%)
END IF

RESTORE Voorbeeld2Data
Request win&,320&,100&

IF errnr%=0 THEN
  LOCATE 21
  IF response&=1 THEN PRINT "Goed!"
ELSE PRINT "Fout!"
ELSE
  PRINT "FoutCode ";STR$(errnr%)
END IF

 PRINT:PRINT "Einde Programma -- Druk
Toets"
 WHILE INKEY$=""
 WEND

END

'--------------------------
' Dataplaatsing VOOR de SUBS voor de
AC-BASIC
' compiler

Alert1Data:
 DATA 6
 DATA 90,24,"Het programma heeft niet
voldoende geheugenruimte"
 DATA 90,34,"ter beschikking! Maakt U
A.U.B. meer geheugen vrij!"
 DATA 90,44,"Start u daarna dit
programma opnieuw en probeer het"
 DATA 90,54,"nog eens...  (Grapje...)
 DATA 180,74,"Druk op een Muistoets"
 DATA 180,80,"---------------------"

Alert2Data:
 DATA 4
 DATA 70,20,"Guru Meditatie #81000009 :
Geheugenplaats 2x vrijgegeven"
 DATA 70,29,"Guru Meditatie #00000003 :
De Guru die iedereen heel erg haat!"
 DATA 20,45,"<LeftButton #81000009>"
 DATA 440,45,"<RightButton #00000003>"

Voorbeeld1Data:
 DATA 5
 DATA 0,1,0,2,10,"  Dit is een
voorbeeld van"
 DATA 0,1,0,2,21," een AutoRequester in
Basic"
 DATA 3,1,0,2,28,"
-------------------"
 DATA 0,1,0,6,3,"LEUK"
 DATA 0,1,0,6,3,"FLAUWEKUL"
```

```
Voorbeeld2Data:
 DATA 7
 DATA 2,1,0,40,10," Met welke
systeemroutine"
 DATA 2,1,0,40,20,"  kunt u zeer
makkelijk
 DATA 2,1,0,40,30,"  een Requester
maken ?"
 DATA 3,1,0,120,43,"-/-"
 DATA 0,1,0,85,54,"(C-Info '89)"
 DATA 0,1,0,6,3,"AutoRequest()"
 DATA 0,1,0,6,3,"BuildSysRequest()"

'Request 11 Juni 1989
'====================
'Syntax: Request
WindowHandle,breedte,hoogte
'De windowhandle van het aktieve window
verkrijgt men door whandle&=WINDOW(7)
'De datapointer behoort op de data te
staan die deze routine nodig heeft
'VB: datapointer op QuitData dan :
RESTORE QuitData
'errorcodes: 1 niet genoeg geheugen / 2
te weinig data
'"Syntax" RequestData:
'  DATA aantalregels,
'  DATA
FrontPen,BackPen,DrawMode,LeftEdge,TopEdge
,"de tekst"
'

SUB Request(win&,sx&,sy&) STATIC
  SHARED errnr%,response& '(*AMG)
  '(*AC) SHARED
errnr%,response&,ACString$()

  ' Reserveer een buffer van 500 bytes
waar de intuitext
 ' structuren ingaan. Deze zijn 20
bytes groot (eigenlijk 19,
 ' maar 20 even). 500 bytes is dus
genoeg voor 500/20=25 regels!
 ' De grootte van het buffer kunt u
natuurlijk ook zelf veranderen.

  mem&=AllocMem&(500,0)
  IF mem&=0 THEN errnr%=1:EXIT SUB

  ' Haal de informatie voor de
intuitext structuren
  memres&=0
  imem&=mem&
  READ lines%

  ' minstens 3 regels : 1 voor de
bodytext en 2 voor de text van het
  ' JA / NEE gadget.

  IF lines%<3 THEN errnr%=2:CALL
FreeMem&(mem&,500&):EXIT SUB

  FOR i%=0 TO lines%-1
    '(*AC) READ
frp%,bp%,drwm%,Le%,Te%,ACSTRING$(i%)
    READ frp%,bp%,drwm%,Le%,Te%,Body$
'(*AMG)

    ' We sluiten de string af met een 0
(ASCII code 0),
    ' omdat de systeemroutines de 0
zien als het einde
    ' van een string.

    Body$=Body$+CHR$(0) '(*AMG)
    '(*AC)
ACString$(i%)=ACString$(i%)+CHR$(0)

    ' We poken de gegevens voor de
intuistructuren
    ' in het buffer.
    POKE imem&,frp%
    POKE imem&+1,bp%
    POKE imem&+2,drwm%
    POKEW imem&+4,Le%
    'een integer of long integer (16/32
bits) moet op een EVEN geheugenplaats
gezet worden
    POKEW imem&+6,Te%
    POKEL imem&+8,0&
    '(*AC) POKEL
imem&+12,SADD(ACSTRING$(i%))
    POKEL imem&+12,SADD(Body$) '(*AMG)
    POKEL imem&+16,imem&+20

    imem&=imem&+20
    'zet de pointer naar de volgende
intuistructuur

  NEXT

    ' De laatste 3 (laatste Regel +
JA/NEE tekst) mogen niet met elkaar
   ' verbonden zijn, dus poken we een 0
(long) op de plaats waar een pointer
   ' voor NextText zou moeten staan.

  POKEL (imem&-4),0&
  POKEL (imem&-24),0&
  POKEL (imem&-44),0&


response&=AutoRequest&(win&,mem&,imem&-40&
,imem&-20&,0&,0&,sx&,sy&)
  CALL FreeMem&(mem&,500&)
  errnr%=0
END SUB

'Alert 11 Juni 1989
'==================
'Syntax : Alert hoogte% (de hoogte van
de rood knipperende box)
'de datapointer dient op de benodigde
data te staan
'VB: datapointer op Geheugendata :
RESTORE Geheugendata
'errorcodes : 1 niet genoeg
systeemgeheugen / 2 te weinig data
'              : 3 te klein buffer
'"Syntax" alertdata:
'  DATA aantalregels,
'  DATA LeftEdge,TopEdge,"de tekst"

SUB Alert(height&) STATIC
 SHARED errnr%,response&
```

```
 'er worden 600 bytes gereserveerd, dat
naar gelang de
 'behoefte verandert kan worden
 bufmemsize&=600
 mem&=AllocMem&(bufmemsize&,0&)
 IF mem&=0 THEN errnr%=1:EXIT SUB
 bufmemleft&=bufmemsize&
 mptr&=mem&

 'lees data in
 READ lines%
 IF lines%<1 THEN errnr%=2:EXIT SUB
 FOR i%=0 TO lines%-1
   READ Le%,Re%,Alert$

bufmemleft&=bufmemleft&-5-LEN(Alert$)
   IF bufmemleft&<0 THEN
     CALL FreeMem&(mem&,bufmemsize&)
     errnr%=3
     EXIT SUB
   ELSE
     POKE mptr&,((Le% AND 768)/256)
     POKE mptr&+1,(Le% AND 255)
     POKE mptr&+2,Re%
     sdata&=SADD(Alert$)
     FOR l%=0 TO LEN(Alert$)-1
       POKE mptr&+3+l%,PEEK(sdata&+l%)
     NEXT
     POKE mptr&+3+l%,0
     IF i%=lines%-1 THEN
       POKE mptr&+4+l%,0    ' END code
     ELSE
       POKE mptr&+4+l%,255 ' NOEND code
     END IF
     mptr&=mptr&+l%+5
   END IF
 NEXT

 RECOVERYALERT&=0&

response&=DisplayAlert&(RECOVERYALERT&,mem
&,height&)
 CALL FreeMem&(mem&,bufmemsize&)

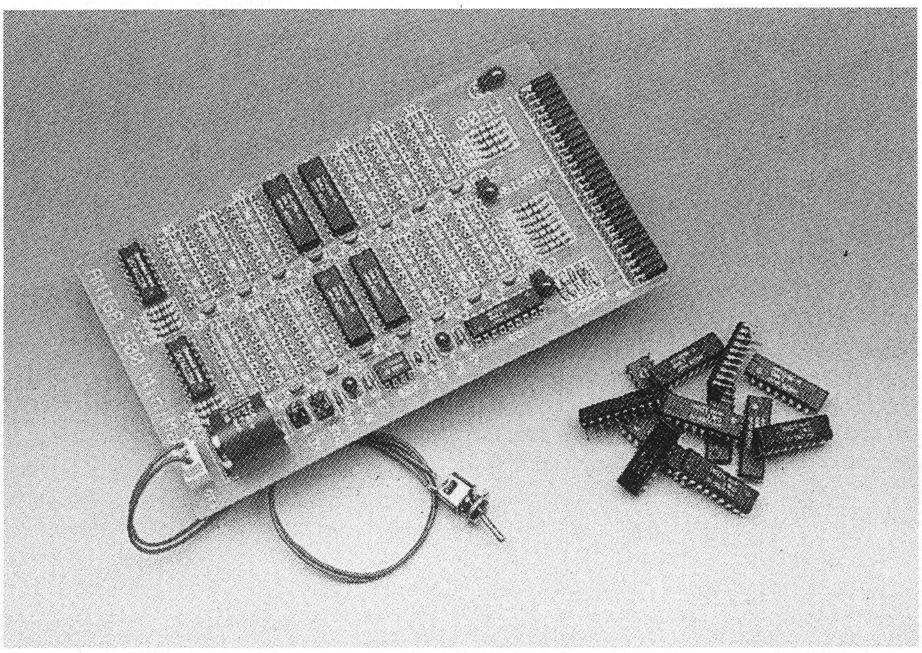
END SUB
```

Einde listing Alert

Amiga RAM-uitbreidingen

Met meer geheugen meer mans

Geheugen kan er eigenlijk nooit genoeg in je Amiga zitten. Niet alleen lopen in het RAM opgeslagen programma's aanmerkelijk sneller dan van disk te laden software, maar ook worden de programmatuur en grafische bestanden steeds groter. Zonder voldoende RAM wordt de Amiga dus al gauw traag en beperkt in zijn mogelijkheden. Ten einde deze "computerdementie" te voorkomen, is de aanschaf van een RAM-uitbreidingskaart al gauw gewenst.



MiniMax, een professionele ram uitbreiding voor de Amiga.

Achterhaald

Weg zijn de tijden waarmee je op een Commodore Pet nog een tekstverwerkertje kon laten lopen. Zelfs de 64 KB van de Commodore 64 (bij de intro nog een indrukwekkende hoeveelheid RAM die volgens de toenmalige experts wel nooit helemaal gebruikt zou worden!) zijn reeds lang achterhaald.
In de begindagen kwam de Amiga 1000 nog met slechts 256 KB aan boord. Nu een lachertje voor een PC. Al snel kwam er 256 KB RAM- uitbreiding op de markt, want anders kwamen de grafische en geluidsprestaties niet tot hun recht. De modernere Amiga 500 beschikt standaard over 512 KB en de Amiga 2000 komt met 1 MB aan boord uit de winkel. Het is allemaal echter nog steeds niet genoeg en RAM-expansies van 2 tot 4 MB lijken nu al eerder regel dan uitzondering te worden.

Waarom meer RAM?

De 680X0 CPU verkeert in een voortdurende communicatie met het **Random Access Memory**. Gegevens uit het RAM worden via de geheugenbus naar de Motorola microprocessor overgebracht en na bewerking weer geretourneerd. De mogelijkheid om via Direct Memory Access (DMA) snel grootte datablokken tussen de harddisk en CPU te transporteren, laten we hier even buiten beschouwing.

Het dataverkeer tussen RAM en CPU is sneller dan de communicatie tussen de microprocessor en de diskettestations of harddisk. Daarom draaien in het RAM staande programma's ook veel sneller dan software die steeds, al of niet in modules, van schijf geladen moet worden. Software in RAM betekent pure verwerkingskracht.
Een tweede probleem is de toenemende **programma-omvang**. Diverse pakketten hebben al twee 3.5 inch floppies of meer nodig om alle informatie te bevatten. Dat betekent in de praktijk werken met twee diskdrives of een harddisk anders ontstaat er een soort 'diskjockeywedstijd' tussen gebruiker en Amiga-programma. Indien het mogelijk is om het gehele pakket in het Amiga-RAM te laden, vervalt de noodzaak steeds met tijdsverlies programmamodules van de schijf te laden.
Gebruikers van harddisks kennen waarschijnlijk de techniek van **disk swapping**. Daarbij worden grote datablokken volgens een bepaald logisch (qua gebruiksvolgorde) algoritme in een apart hard-disk-bestand gezet. Het swapfile wordt in zijn geheel tussen het RAM en de harde schijf uitgewisseld op het moment dat de software de desbetreffende data nodig heeft. Deze truuk bespaart flink op de zoektijd, maar vormt verder slechts een surrogaat voor een echte RAM-expansie.
Multi-tasking is een derde RAM-vreter. Meerdere programma's tegelijkertijd draaien betekent dat elk pakket afzonderlijk ook recht op een eigen stukje RAM moet hebben. En dan slinkt de RAM-koek snel af. Voor een goede multitasking zijn minimaal 2 MB aan vrij RAM nodig.
Behalve de programma's groeien ook de werkbestanden steeds verder uit. Gedetailleerde DTP-documenten, met teken pakketten ontworpen artwork, CAD/CAM, animaties, gedigitaliseerde videobeelden en geluid/muziek-files noem maar op; één MB is zo vol. Deze grote files nemen zelden genoegen met diskswapping en willen een echt vrij **Chip-RAM**.

Paging-technieken zijn een methode om het RAM in geheugenbladeren op te delen. Terwijl de CPU naar de ene pagina kijkt, kan de andere vervangen of ververst worden. Ook deze geheugenpagina's worden met name bij sommige video- en animatietoepassingen steeds groter, zodat de RAM-behoefte navenant stijgt. Overigens is de paging een voor de Amiga achterhaalde en te trage PC-techniek. De Fat Agnus is immers in staat om met 8 MB aan intern RAM te werken.
Het gebruik van steeds hogere videoresoluties doet een zware aanval op het beschikbare RAM. Ook hier is de groei er nog lang niet uit.
Last but not least het gebruik van snellere CPU's. Door toepassing van MC 68020- en MC 68030-turbokaarten stijgt de honger naar meer snelle RAM-banken aanzienlijk. Het is zelfs zo dat u de prestaties van de turbo-kaarten niet optimaal kunt benutten zonder afdoende RAM-voorzieningen.

RAM-aspecten

Het Amiga-RAM bestaat uit **geheugenchips**. Die zijn er in alle soorten en maten met elk hun specifieke eigenschappen. En dan is er nog de Amiga-hardware zelf, die op een of andere manier met de geheugenexpansie moet kunnen samenwerken. Eerst de **chips zelf**. Daarbij krijgt u te maken met:

- De **hoeveelheid RAM in één chip**; Er zijn bijvoorbeeld RAM-chips van 256 KB en 1 MB. Van het laatste type zijn er vanzelfsprekend minder nodig. Verder is er nog de zogenoemde SIMM, Single In line Memory Module, een soort snelle super geheugenchips, waarmee u snel een geheugenkaart tot vele MegaBytes kunt uitbreiden
- De **snelheid van de RAM-chips**; De geheugeninformatie in de chips dient voortdurend cyclisch ververst en onderhouden te worden. Dat kost systeemtijd, want tijdens deze processen kan de CPU de desbetreffende RAM-bank niet gebruiken. De snelheid van de RAM-chips wordt opgegeven in nanoseconden (ns). Hoe minder ns des te sneller het geheugen. In de praktijk onderscheidt men **Fast-**, **Dynamic-** en **Static-RAM**. De DRAM-chips zijn het goedkoopst en traagst. SRAM-chips zijn veel sneller en behoorlijk duurder.
- **Hoeveel RAM** past er eigenlijk op de beoogde RAM- uitbreidingskaart? Nu lijkt 2 MB wellicht voldoende, maar straks wilt u misschien wel 4 of zelfs 8 MB. Bij gebrek aan Amiga-slots is alles op één enkele kaart een must.
- Het **type-RAM**. Meestal is het slechts mogelijk om een RAM-bank op de uitbreidingskaart met hetzelfde type chips uit te vullen. Andere pootjes, snelheid of grootte geven geheid ernstige storingen. Zet daarom alleen dezelfde typenummers in de bank.
- De **hardware van de uitbreidingskaart** bepaalt met welke RAM-chiptypen er gewerkt kan worden. Verder heeft de geheugenbus 16/32-bit van de RAM-expansiekaart een flinke snelheidsvinger in de pap.
- De geheugenkaart dient **compatibel** met de overige Amiga- hardware te zijn. Bijvoorbeeld niet in conflict te raken met een PC-emulatiekaart.
- **Autoconfiguratie** en de mogelijkheid tot **tijdelijk uitschakelen** voorkomt veel praktijkproblemen.

Het zal duidelijk zijn dat je als verstandige Amiga-gebruiker niet zomaar eventjes naar de computershop holt om de eerste de beste geheugenkaart uit het schap te trekken. Eerst duidelijk inventariseren waar de RAM-kaart voor bedoeld is, hoe snel deze dient te zijn, welke chips er op moeten en wat het allemaal gaat kosten. Dat voorkomt later veel ellende. Gelukkig behoort softwarecompatibiliteit bij de moderne Amiga-software tot het verleden.

RAM-gebruik door de Amiga

De Commodore Amiga deelt het RAM in drie verschillende gebruikstypen in. Als eerste het zogenaamde **Chip-RAM**. Dit is het gewone gebruiks-RAM waar al of niet via DMA-technieken de datafiles voor

Merk/type	voor model Amiga	aantal MB's tot maximaal	autoconfiguratie	diversen	Fabrikant	prijs (1,8 of 2 MB)
Combitec DRAM 2000	2000	2 , 4 of 8 MB	ja	0 wait states SIP	Combitec	*f* 1400,-
Microbotics 8-Up	2000	2, 4, 6 of 8 MB	ja	0 wait states met 1MB SIMMs	Microbotics	*f* 1400,- *f* 1550,-
A 2058	2000	2, 4 of 8 MB	ja	1 MB-chips	Commodore	*f* 1400,-
Impact A 2000	2000	2 MB	ja	SCSI-controler/RAM	GVP	*f* 1700,-
Pro-RAM 2000	2000	2, 4, 6 of 8 MB	ja	0 wait states	Progressive Periherals & software	*f* 1100,-
Jocheim 2000 RAM	2000	2, 4, 6 of 8 MB	ja	veel PAL-bouwstenen	Jochheim Computer Tuning	*f* 990,-
A 502	500	512 KB!	ja	alleen 512 KB	3-State Computer technik	*f* 220,-
Combitec DRAM	500/1000	2, 4 of 8 MB	ja	harddisk-adapter	Combitec	*f* 1200,-
Golem RAM-Box	500/1000	2 MB	ja	uitschakelbaar	Golem/Kupke	*f* 1100,-
Pro-RAM	500	512 KB of 1,8 MB	ja	uitschakelbaar, intern	Intelligent Memory GMBH	*f* 850,-
Minimax	500	512 KB of 1,8 MB	ja	intern	Gigatron	*f* 880,-
EXP-1000	500	1 MB	ja	extern	Progressive Pheripherals & Software	*f* 660,-

Overzichtstabel RAM-kaarten Amiga 500, 1000 en 2000/2500

audio, video, databases, tekstverwerkers, spreadsheets en dergelijke worden gedumpt. Bij bemoeienis van de 68000 CPU werkt het Chip-RAM relatief traag. Als de CPU met het RAM bezig is, liggen andere taken immers tijdelijk stil.
Bij de oudere Amiga-typen uit de pré-Enhanced Chip-Set-periode kan de **Fat Agnus**-besturingschip slechts 512 KB vrij Chip- RAM tegelijk aan. De nieuwe Fat Agnus-telg (volgens insiders de Fang) kan nu met 1 MB aan vrij Chip-RAM uit de voeten. Dat betekent het einde van de Memory Overflow-problemen bij programma's die te grote video- en audio-files in het Chip-RAM proberen te proppen. Het **Fast-RAM** is een extern gedefinieerd RAM, dat zo'n 80% sneller is als het conventionele Chip-RAM. Maximaal staat 8 MB aan Fast-RAM ter beschikking dat niet direct voor de video/audiochips en DMA-technieken toegankelijk is. Voor het gebruik van dit snelste Amiga-RAM gelden speciale regels. Zie hiervoor uw Amiga- of RAM-expansiekaartmanual.
Het zogenaamde **Ranger Memory** (of intern Fast-RAM) is een wat vreemde eend in de geheugenbijt. Ook hier is het geheugen niet direct toegankelijk voor de video/audio-chips en DAM-technieken. De snelheid valt te vergelijken met die van het Chip-RAM. Bij de Amiga 2000 model A staat maximaal 1 MB Ranger Memory tot de beschikking van de gebruiker. Een model B met de nieuwe Fat Agnus kent slechts 512 KB dia via jumpers ook als Chip-RAM geconfigureerd kunnen worden.
Informeer bij uw leverancier om welk type Agnus het gaat alvorens het geheugen te configureren of een RAM-uitbreidingskaart te kopen. Anders kunt u rare dingen beleven!

De in/uitbouw

Bij de Amiga 500 en 1000 zijn de inbouwmogelijkheden beperkt. Een Amiga model 1000 biedt aan de voorzijde een geheugen uitbreidingsslot, waar zover ons bekend, maximaal 1,8 MB aan RAM-uitbreiding in passen. De techniek staat echter niet stil en wie weet heeft iemand indmiddels een 4 MB-oplossing bedacht. Een andere mogelijkheid voor het **model 1000** is RAM-uibreiding via de expansiebus. Behalve losse RAM-kastjes zie je tegenwoordig steeds meer integratie van extra RAM in diskdrive- en/of harddiskmodules. In die uitbreidingskastjes kan de gebruiker een extra RAM-kaart steken.
De **Amiga 500** was van Commodore-huis uit al berekend op uitbreiding tot 1 MB. Een aantal leveranciers doet bij die 512 KB RAM-uitbreiding ook nog eens een leuk systeemklokje. Momenteel zijn er voor de Amiga 500 al interne uitbreidingen tot 4 MB te koop. Extern kan ook net als bij de Amiga 1000 via de expansiebus en integratie met de drive-uitbreiding.
Bij de **Amiga 2000** (model A en B) gaat het in de praktijk om RAM-uitbreiding via één van de nog vrije Amiga-slots. Daarin passen auto-configurerende RAM-expansiekaarten. Let er wel even op welke Fat Agnus er precies in de machine zit. N.B.: Commodore heeft ECS-upgrades voor de geleverde A-modellen aangekondigd. Daarmee zijn ook de voor model B ontworpen geheugenkaarten bruikbaar.

Meer geheugen is een must. Anders kan je het voor de Commodore Amiga's 500, 1000 en 2000/2500 niet noemen. De wens tot hogere systeemprestaties, de steeds groter wordende programmatuur en datafiles, hogere schermresoluties en het gebruik van turbo- kaarten overgroeien de standaard 512 KB of 1 MB van uw Amiga. Om de prijs behoeft men het eigenlijk niet meer te laten. De RAM- chips kosten steeds minder en de lege kaarten zelf kosten nog iets van *f* 300,- tot *f* 400,-. Volgens sommige postorder- advertenties zet de Amiga-gebruiker al voor minder dan *f* 1.000,- 2 MB extra in zijn/haar machine.

U.S.

Amiga starters



Een nieuw artikel voor beginnende Amiga bezitters. We hopen dat beginners dit artikel goed kunnen gebruiken bij hun eerste, verwoede pogingen om wat zinnigs te doen met hun nieuwe aanwinst.

Even een hypothetische situatie, een gebruiker, sorry AMIGAGEBRUIKER, zet z'n Amiga voor het eerst aan en ziet een leuk symbooltje op het scherm verschijnen. Alles goed en wel maar wat hier te doen?
De oplossing is eenvoudig genoeg, pak dat schijfje wat GRATIS! meegeleverd is (voor Nederlanders zeer belangrijk), en kijk of de diskette beveiligt is. Zo niet, doe dit dan door het kleine schuifje linksboven aan de achterkant van de diskette, helemaal naar beneden te schuiven. Nu kunt u door het gaatje heen kijken. De diskette is als gevolg hiervan niet meer beschrijfbaar. Als u later toch informatie of veranderingen van de Workbench wilt bewaren op deze diskette, dan raden we u aan eerst een kopie van de Workbench te maken, en deze voortaan te gebruiken. Op deze wijze blijft de originele Workbench diskette onveranderd. Hierdoor heeft u altijd nog een originele versie achter de hand. Nu kunt u rustig verder gaan met de beveiligde diskette. Later leggen we wel uit hoe u deze schijf kunt kopieëren. Stop nu de diskette in de ingebouwde drive van de computer. Deze drive wordt aangeduidt als *df0:*. Vele programma's verwijzen op deze manier naar de interne drive. Zou er staan drive *df1:* dan bedoelen ze hier mee de tweede drive die op de computer aangesloten is of aangesloten kan worden.

Beschadigingen voorkomen

Nu U de Workbench schijf in de drive heeft gedaan ziet u een groen lampje branden, wat inhoud dat de computer op de schijf informatie aan het lezen is. Onthoudt nu dat u *NOOIT* de diskette uit de drive haalt, aangezien u daardoor de drive en de diskette kunt beschadigen. Na een tijdje wachten ziet u op het scherm een afbeelding van de Workbench. De Workbench-diskette wordt gesymboliseerd door een tekeningetje (ikoon) rechtsboven op het beeldscherm. Activeer deze ikoon met een klik van de linkermuisknop. Bovenaan het scherm staat nu in de titelbalk dat u met de workbench ikoon aan de gang bent. Er zijn hedentendage twee versies van de Workbench te weten versie WB 1.2 of de nieuwere versie 1.3. Wij zullen ons verder bezighouden met de 1.2 versie, aangezien de meeste Amiga bezitters nog niet in het bezit zijn van de nieuwste 1.3 versie.
Nu kunt u op twee manieren de Workbench ikoon starten, (dit geldt ook voor andere ikonen).

- 1. Doormiddel van een dubbele klik op de workbench ikoon.
- 2. Door na het activeren van een ikoon, door een enkele druk van de linker muisknop, de rechtermuisknop ingedrukt te houden, de zogeheten menu toets, en boven aan het scherm naar de nu verschenen titelbalk te sturen. Kies de optie 'Open'.

Nog even terugkomende op optie 2. In de titelbalk zijn een aantal opties kiesbaar. Een drietal menutitels zijn mogelijk, namelijk Workbench, Disk en special. Zonder de menutoets los te laten, wijst u de titel 'workbench' aan. Nu valt er een lijst-

je uit de balk (schrik niet, blijf recht op zitten, en houdt de rechterknop ingedrukt). Wijs nu 'Open' in de lijst aan. 'Open' licht nu op. Om 'Open' te kiezen laat u gewoon de rechterknop weer los. Zo opent U de Workbench diskette.
Dit lijkt allemaal ingewikkeld maar het stelt niks voor, het is gewoon een handigheid die je even door moet hebben.

Venster

Op het beeldscherm verschijnt nu een venster genaamd A500 WB 1.2. Hierin staan 8 ikonen namelijk Demos, Utilities, System, Expansion, Empty, Preferences, Clock, en Trashcan. Deze ikonen kunt u verslepen. We zullen dat aan de hand van een voorbeeld demonstreren.

Verslepen van ikonen en vensters

U gaat met het pijltje naar het ikoon RAM disk, rechtsboven bij het Workbench ikoon, en opent deze op dezelfde manier als hoe u het Workbench ikoon opende. Nu verschijnt er opnieuw een venster, genaamd RAM disk. Om nu het Workbench venster ook nog te kunnen gebruiken moeten we eerst het RAM disk venster verplaatsen (verslepen). Dit doen we door naar de bovenste balk van het RAM disk venster te gaan, de zogenaamde sleepbalk, de linkermuisknop in te drukken en vast te houden, en zo het venster te verslepen naar een willekeurige plaats. En ja, daar komt het Workbench venster ook weer te voorschijn. Laat nu de linkerknop weer los en klaar is Kees. Op deze manier heeft u al twee vensters tot uw beschikking.
Pak nu een ikoon uit het Workbench venster. Als voorbeeld gebruiken we het Clock ikoon. Ga met het pijltje naar de Clock, en druk op de linkermuisknop die u vast houdt. Sleep het ikoon naar het RAM disk venster en laat het daar weer los. U ziet, niet alleen vensters maar ook ikonen kunnen versleept worden.
Nu gaat het groene lampje weer branden, dit als teken dat de computer het ikoon Clock laad en het in de RAM disk plaatst.

Snelheid van de RAM disk

U moet de Ramdisk zien als ramgeheugen dat als een diskdrive gebruikt kan worden en waar snel informatie uitgehaalt kan worden, sneller als van diskette.
Dit kunnen we illustreren door een nieuw voorbeeld. We laden het ikoon 'clock' in het Workbench venster, dus niet die in het RAMdisk venster staat. Dit doen we doormiddel van een dubbele klik met de linkermuisknop. Na zo'n 4 a 5 seconden wachten verschijnt er een nieuw venster, dit is het clock venster. Deze bespreken we later uitgebreider. Voor nu halen we eerst dit venster weer weg, door met de linkermuisknop op het sluitgadget te drukken. Deze zit linksboven in het clock venster. Bijna alle vensters hebben zo'n sluitgadget en dat is maar handig ook, want met veel vensters tegelijk op het scherm, werkt de computer ook niet meer op optimale snelheid!
Nadat dit gebeurt is, laden we de clock opnieuw, alleen nu vanuit het RAM disk venster. U ziet dit gaat stukken sneller dan vanuit het workbench venster. In ongeveer 1 a 2 seconden heeft u al beschikking over deze utility.
Onthoudt echter goed, de RAMdisk kost veel geheugen. Omdat alles dus niet, zoals het woord disk zou zeggen, op een diskette (schijfje) wordt geplaatst maar rechtstreeks in het geheugen van de computer wordt gezet. Het is alleen maar handig als u regelmatig gebruik maakt van zo'n ikoon, omdat u anders alles net zo goed in een keer had kunnen laden vanuit de Workbench en zodoende het RAM geheugen kunt sparen voor andere doeleinden.

Voordeel en nadeel RAMdisk

Als u vele aantekeningen zou moeten maken kunt u het beste de Notepad, die in het Workbench venster in het utility ikoon staat, op de RAM disk zetten. Het voordeel is dat U dan snel en gemakkelijk weer een aantekening kunt maken met de Notepad. Maar er is wel een nadeel aan de RAM disk verbonden. Zoals net hierboven beschreven is het geheugen op deze manier snel vol. En na een reset van de computer is alles wat u in de RAM disk heeft ingevoerd verloren gegaan. Zodoende moet u regelmatig ingevoerde gegevens op de diskette opslaan. Anders zou uren werk verloren kunnen gaan, en dat is natuurlijk niet de bedoeling.

Clock

Even terugkomen op de clock. Dit is een bijzonder handig programma. Zo handig dat in sommige bedrijven stagaires worden verplicht een dergelijke utility voor een PC te schrijven. Dat even terzijde, bij de Amiga is het al aanwezig. Naast het aangeven van de tijd kan 'clock' ook als wekker gebruikt worden. Laad de clock opnieuw vanuit de RAM disk (lekker snel) en u ziet nu opnieuw de clock verschijnen.
Dit is een analoge clock, met sekondenwijzer en datum geplaatst in een venster. Om nu veranderingen aan te brengen moet u eerst het clock venster activeren. Dit doet u doormiddel van een klik met de linkermuisknop in het clock venster. Nu veranderen de letters Clock V2.15 van gestippelde in normale letters. Dit als teken dat het venster is geactiveerd. Dit is hetzelfde als wat u ook al bij de Workbench in het begin heeft uitgevoerd. Daardoor kunt u nu uit de menubalk een keuze maken met betrekking tot de clock. Druk de rechtermuisknop in, (vasthouden) en voila! daar verschijnt de menubalk. De keus is uit Type, Mode, Seconds, Date en Alarm. Ga als voorbeeld naar het woord Type, en laat de desbetreffende inhoud naar beneden kletteren. Deze bestaat uit Analog, Digital 1 en 2. Kies als voorbeeld Digital 1. Nu verdwijnt de analoge klok en komt er een digitale te voorschijn. Maak nu van de 12 hour aanduiding een 24 hour aanduiding. Dit kunt u doen vanuit de menubalk keuze mode (24 hour). Ook kunt u de datum laten verdwijnen en het alarm instellen. Probeer dit maar eens.
Als u de tijd van de klok wilt veranderen, zult u dat moeten doen vanuit de Preferences. Dit ikoon staat in het eerste venster (het Workbench venster). We komen straks nog terug op de Preferences. We zullen dan uitleggen wat men hiermee al-



lemaal kan doen. Laten we nu terugkeren naar de analoge klok, en met dit venster nog wat grapjes uithalen.

Vergroting vensters

Laten we proberen het venster van de klok te vergroten. Dit doen we door rechtsonder naar het scherm te gaan, en de omvanggadget (Sizing Gadget) in te drukken, en te verslepen tot de gewenste grootte is bereikt. Nu heb je dus kans dat het Workbench venster niet meer te lezen is, echter ook hier is aan gedacht.

Vensters terug "toveren"

Het is mogelijk doormiddel van twee gadgets, de twee rechtsboven in het venster, het venster voor of achter een ander venster plaatsen. Met het linker gadget kun je het scherm, doormiddel van een druk met de linker muisknop, voor een ander halen. Met het rechter gadget kun je het scherm weer achter een venster plaatsen. Probeer dit!
U zag, het klinkt moeilijk, maar het is eigenlijk een 'peace of a cake'.

Scrollen van de vensterinhoud

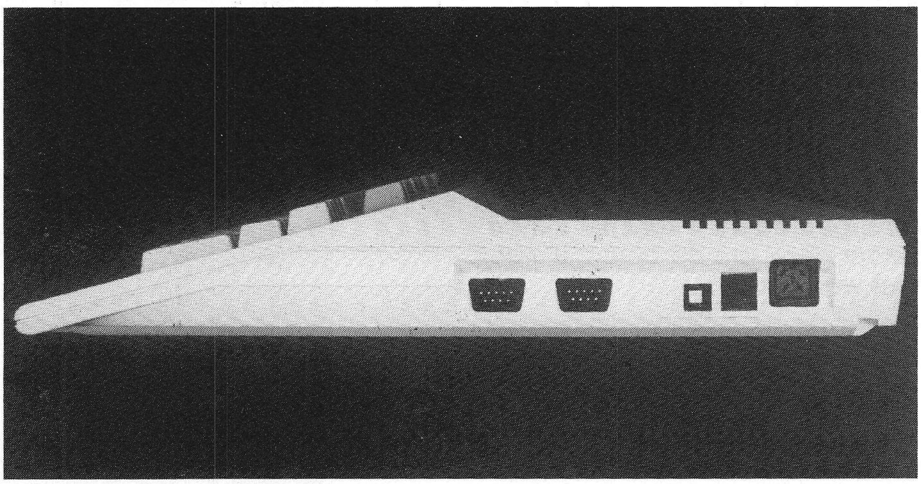
Aangezien het Workbench venster lekker vol staat, kun je het hier het beste op (of in) (of mee) proberen. Dus haal voor deze mogelijkheid het Workbench venster op de voorgrond, of haal de clock weg door op het sluitgadget te drukken.
Verklein nu het Workbench venster zo, dat niet alles er meer inpast. Nu kunt u doormiddel van de pijltjes, die in de rand van het venster zitten de ikonen in het scherm laten scrollen. Als het scrollen u te snel gaat kunt u bij het selecteren van een scrollpijl de SHIFT-toets indrukken, hierdoor scrollt nu de inhoud van het venster per pixel over het scherm.
Ook kunt u nu aan de balk, die tussen de lijntjes staat, aflezen hoe vol het venster werkelijk is en wat u nu ziet. Zodoende kan een venster zo groot uitvallen dat het niet in zijn geheel op het scherm past maar u toch alles heel makkelijk kunt bekijken door die ene druk van de linkermuis knop.

Diskettemeter

Als laatste handigheid van een venster behandelen we de diskettemeter.
Deze balk is oranje en zit aan de linker kant van het venster. Hierop kunt u aflezen hoe vol de diskette staat, dus hoeveel bytes erin beslag genomen zijn door programma's die op de schijf staan. Let erop, als de balk helemaal tot bovenaan staat, dus bijna tegen de bovenrand van het venster, dan zal de schijf in z'n geheel volstaan. Het zal dan ook niet lang meer duren voordat de medeling 'Disk full' verschijnt. Probeer dan nog eens 'retry' in te drukken. In vele gevallen zal de schijf nog niet helemaal vol zijn!

Preferences

Nu zullen we terugkomen op het eerder genoemde preferences programma. Met behulp van dit programma kunt u een aantal veranderingen in uw Amiga aanbrengen. We zullen nu de meest gebruikte mogelijkheden van preferences aan u uitleggen.
We vinden het preferences ikoon in het Workbench venster. Zoek het ikoon op en selecteer dit. Er verschijnt nu een venster. Zou u dit venster nu vaak gebruiken als u met de Workbench bezig bent, dan is het handiger dat u dit ikoon in de RAM disk plaatst.



Datum en tijd veranderen

Dit gaat heel simpel. U kiest het getal dat u wilt veranderen en verhoogt of verlaagt dat naar eigen wensen. That's all.

Centreren van het scherm

Met de meeste monitoren kunt u zelf het beeld op de perfecte plaats laten verschijnen (centreren). Met preferences kunt u dit ook doen. Alleen geldt het nu natuurlijk voor de Workbench, en niet voor andere programma's.
U centreert met het hoekige symbooltje, wat in het midden van het venster staat, in de Display Centering Gadget. Beweeg de muis en zie het scherm verschuiven.

Toetsherhaling

Als u de snelheid hiervan wilt veranderen doet u dat ook met preferences.
Ga met de pijl naar Key Repeat Speed en versleep het rondje naar de gewenste snelheid.
U zult nu ongetwijfeld denken, 'wat heb ik hier nou aan', aangezien we alleen nog maar de muis gebruikt hebben. Toch, als u wat verder opweg bent met de Workbench, zult u later ook het CLI gaan gebruiken. En dat IS, vrijwel alleen maar typewerk. Voor meer informatie over de CLI bekijk hiervoor de *Binnenin Amiga-DOS* cursus afleveringen.

Wachttijd voor begin van toetsherhaling

Na het indrukken van een toets, is er een wachtperiode voordat het herhalen van deze toets begint. Deze wachtperiode kunt u regelen door het rondje van de Key Repeat Delay balk te verslepen. Short of naar keuze een Long period.

Muissnelheid

Ook de muis is in zijn snelheid te regelen. Dit door middel van drie muissnelheidsinstellingen. De instellingen 1, 2 en 4 geven het aantal centimeters aan dat de muis bewogen moet worden om de pijl ongeveer een derde van de schermbreedte te laten afleggen.

Tijdsinterval voor dubbele klik

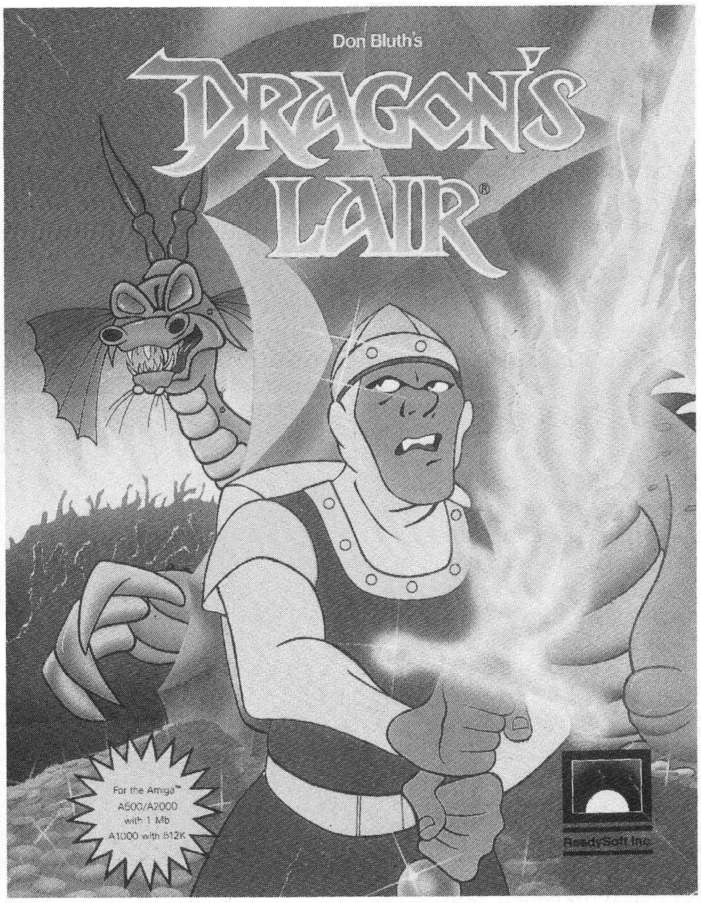
Als u een dubbele klik moet geven om bijvoorbeeld een ikoon te laden dan moet dat gebeuren binnen een bepaalde tijd.
Deze tijd kunt u ook regelen.
Sleep het rondje omhoog om het tijdsinterval te verlagen (dus snel klikken).
Sleep hem naar beneden om de tijdsinterval te verhogen. U heeft nu wat meer tijd om een dubbele klik te geven.

Tot slot

Het lijkt bijna wel een nieuwe aanpak van de Binnenin AmigaDOS cursus. Toch is dit een geheel nieuwe cursus, vanuit een geheel nieuwe invalshoek, namelijk de beginner. De volgende keer meer, te beginnen met het vervolg van de Preferences. We'll be back, bye bye!

Johan & johan.

Dragons Lair



Een sprookjes wereld op de Amiga, realistische beelden gekombineerd met huiveringwekkende geluiden. Dit programma komt het beste tot zijn recht als de Amiga aan wordt gesloten op een versterker met een behoorlijk ver opengedraaide volume knop. (Maar denk om de buren, U weet niet wat U hoort)

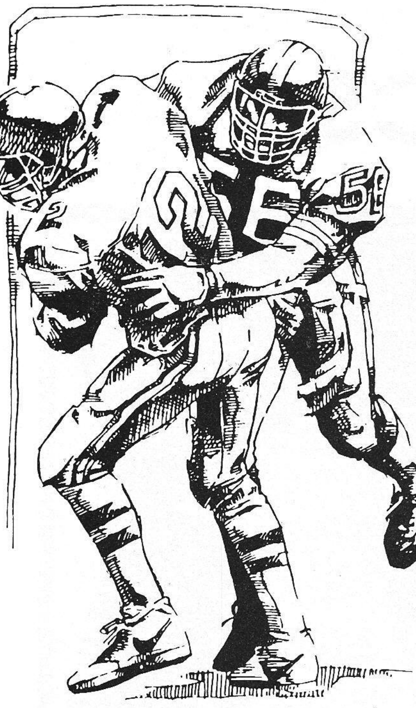
DragonsLair is een spel dat op een geweldige manier gebruikt maakt van de mogelijk- en onmogelijkheden van de Amiga. Grafische beelden worden door muziek begeleid, op zich niet nieuw, maar de manier waarop dit gebeurt maakt het spel zo uniek. Dit heeft ook wel wat nadelen, het gehele spel staat op acht (!!!) diskettes, dus regelmatig moeten er diskettes worden gewisseld. Om het programma op een Amiga te laten draaien is er een geheugenruimte van minstens 1 Mb. noodzakelijk. Een uitzondering hierop is de Amiga 1000, hier wordt door het uitschakelen van de kickstart, voldoende geheugen vrijgemaakt om het spel te kunnen spelen. Al heeft dit wel konsekwenties voor de uitvoering, soms wordt er midden in een beeld even gestopt om te laden, maar een kniesoor die hier op let. Harddisk gebruikers kunnen het spel, door een speciale optie op hun harddisk installeren. U moet er wel om denken dat het dan ook 10 Mb aan (kostbare) diskruimte gebruikt.
Het programma wordt geladen door diskette 1 in de diskdrive te plaatsen als er op het scherm het plaatje van de workbech verschijnt. Het opstart scherm met daarop alle spel gegevens verschijnt na ongeveer 15 seconden op het beeldscherm. Het programma herkent een eventueel aangesloten 2e diskdrive. U kunt de tweede diskette dan in de tweede diskdrive plaatsen. Op het moment dat er van drive twee wordt geladen kan dus disk 3 in de eerste diskdrive. Hierdoor is het mogelijk het spel sneller te laten verlopen, er hoeft niet gewacht te worden met het wisselen van de diskettes. LET OP, nooit de diskette wisselen als het lampje van de diskdrive nog brand, dit kan de diskette onherstelbaar vernielen. Omdat het maken van een backup van dit programma nagenoeg niet mogelijk is, is dan het leed niet te overzien.

Het spel

U moet zich bij dit spel verplaatsen in de huid van de hoofdrol speler, Dirk de Daring. De bedoeling van dit spel is het redden van de prinses Daphne. Zij wordt in een sprookjes achtig kasteel bewaakt door, hoe kan het ook anders, een kwaadaardige, vuurspuwende draak Singe. Om dit geheel nog wat moeilijker te maken zijn er onderweg ook nog vele hindernissen en gemene vallen te ontwijken. Het spel moet gezien worden als een interaktieve tekenfilm. Wat bedoelen we hier mee: Op het beeldscherm speelt zich een verhaal af, als we niet ingrijpen dan is het heel snel afgelopen. En hiermee komen we op de kracht van dit programma, we kunnen de loop van dit spel beïnvloeden. Dit doen we door gebruik te maken van ons numeriek toetsenbord (8 omhoog, 2 naar beneden, 4 naar links en 6 naar rechts, of van de joystick. Het zwaard van onze held wordt getrokken door het indrukken van de vuurknop of het indrukken van de 0 op het toetsenbord.
Het spel wordt bepaald door het indrukken van de juiste toetsen. Nu is niet alleen de juiste toets, maar zeker zo belangrijk is het juiste moment om van de toetsen gebruik te maken. Iets te laat, of iets te vroeg, dit leidt onherroepelijk tot een vroegtijdig einde van onze super held. Omdat er drie levens tot zijn beschikking staan is dit niet direkt het einde van het spel. Om dit 'sterven' weer wat aantrekkelijker te maken wordt dit einde elke keer weer omlijst door een speciale sterfscene met aangepaste geluiden.
Om een idee te krijgen van de juiste richting die moet worden gekozen, zijn er korte lichtflitsen te zien. Probeer deze hints te gebruiken, maar laat U niet alleen hierdoor leidden, gebruik ook het gezonde verstand. Niet alles is wat het lijkt. Het juiste moment waarop het zwaard moet worden getrokken is als het lijkt alsof je door de draak wordt verslonden. En denk erom, te laat, maar zeker ook te vroeg betekent het einde. Er zijn ook situaties waarbij juist niet moet worden bewogen om de hindernis te kunnen passeren. Ervaring is de beste leidraad. Het spel is opgebouwd uit een serie onderdelen, hindernissen, die moeten worden doorlopen. Na elke doorlopen scene wordt het punten aantal verhoogt.

Opties

Er zijn een aantal extra mogelijkheden ingebouwd om de uitvoering van dit spel te



beinvloeden. Het geluid, we kunnen het ons haast niet voorstellen dat U dit wilt, kan worden uitgezet door de "A" in te drukken. De "L" schakelt een filter aan of uit, deze optie moet U proberen, wat geeft het beste resultaat. Iets zeer speciaals geeft het gebruik van de letter "H", hiermee wordt de High Resolution aan of uit gezet. In de High resolution mode is het beeld gecentreerd in het midden gezet. Dit geeft een prachtig scherp beeld, jammer is het alleen dat er dan geen geluid meer bij is. De "I" schakelt de Interlace mode aan of uit. De interlace mode verdubbeld het aantal beeldlijnen. Een nadeel hierbij is dat het bij een (groot) aantal monitoren een flikkering op het beeldscherm geeft.

Einde spel

Er zijn drie manieren om het spel te eindigen. De eerste is de minst aangename, U vindt er niets aan en reset de computer. We kunnen ons dit van een rechtgeaarde Amiga-fan niet voorstellen en gaan naar de tweede manier om het spel te beëindigen. U komt voor grote problemen te staan en heeft alle levens gebruikt. Eén troost, U kunt het spel opnieuw starten, en proberen om het op de leukste manier te eindigen n.l.: U heeft alle problemen kunnen oplossen de draak is verslagen en de schone prinses is gered.

En ze leefden nog lang en gelukkig.......

TV Sports Football

Voor voetbal moesten we tot nu toe twee elftallen, een speelveld, twee doelen en als het even kan wat mooi weer hebben. MirrorSoft heeft dat wat andere gedachten over, een Amiga en het software pakket TV Sports football en een heleboel vrije tijd, zie hier het nieuwe (Amerikaans)voetbal concept.

Wat is het

Het programma is een grafisch arcadeachtig aktie spel. Het kan met 1 of twee spelers tegen de computer worden gespeeld. Competitie spelen, oefenen alles is mogelijk. Teams kunnen zelf worden samengesteld uit een groot aantal componenten. Een extra voordeel is het dat er een groot aantal Amerikaanse termen zijn die je je spelenderwijs eigen maakt. (als het goed is)

Het spel

Wat is de bedoeling van dit spel? Kortweg gezegd meer punten maken dan je tegenstander doet. Om dit geheel in goede banen te leiden zijn er natuurlijk een (groot) aantal regels. Gescoord kan er worden als de bal naar de eindzone wordt gebracht en hier tegen de grond wordt gewerkt. Dit wordt in goed Amerikaans genoemd "Touchdown". Dit levert 6 punten op. Het aanvallende team heeft vier kansen om 10 yards afstand te overbruggen. Hierbij mag de tegenpartij pogingen ondernemen dit te voorkomen. Worden de 10 yards niet gehaald dan draaien de rollen om en wordt de defensieve partij de aanvallende en andersom. Ieder spel wordt verdeeld in een eerste (first) en tweede second) helft, waarbij elke helft wordt verdeeld in een kwart. Bij elke helft wordt de bal geschopt vanaf het startpunt. Een veldgaol kan vanaf elk willekeurig punt op het veld gescoord worden door de bal tussen de palen door te spelen. Dit levert 3 punten op. Zelfs kan er door de defensieve partij worden gescoord. Dit kan de partij dan twee punten opleveren. Er zijn een groot aantal formaties, mogelijkheden om een bal te kunnen spelen. De gebruiksaanwijzing die bij het programma is bijgesloten geeft een groot aantal of- en defensieve patronen aan. Bij alle formaties wordt aangegeven wat in welke situatie de beste oplossing is.

Configuratie

Voor dat het spel wordt gestart is het verstandig van disk 2 een veiligheids kopie te maken. Dit omdat er op deze disk partijen worden weggeschreven en de Amiga kennende er kan hierbij nog wel eens wat mis gaan. Disk 2 is dus niet beveiligd tegen kopiëren.
Noodzakelijk is een Amiga 500-1000-2000 met minstens 512 K geheugen, en er moet ge'boot' zijn met kickstart 1.2. Wanneer U beschikt over twee diskdrive dan plaatst U de diskette REEL 1 in de interne diskdrive (df0) en REEL 2 in de externe diskdrive (df1). Als U het spel al een aantal malen heeft gespeeld is het ook prettig dat U de (overigens indrukwekkende) intro kunt overslaan door het indrukken van de vuurknop op de joystick

Het spelen

Er kan in het menu worden gekozen uit 4 verschillende spelvarianten. Als U gekozen heeft voor de optie EXHIBITION, Moet er nu worden gekozen uit een lijst met tegenstanders. Als U voor één speler heeft gekozen dan is de eerste keuze die wordt gemaakt van dat team dat zelf wordt bestuurd, de tweede keuze is de tegenstander. Bij twee spelers is een tweede joystick in poort twee noodzakelijk. Let op bij de keuze TEAMMATES spelen twee menselijke spelers samen tegen de computer. Joystick 1 is de offensieve de tweede joystick beheerst de defensieve captain, dit geheel geeft het spel een extra dimensie. De LEAGE optie geeft U altijd de keuze uit 28 verschillende teams, welke altijd bestuurt kunnen worden door de mens of door de computer. Zelfs is het mogelijk een eigen team te creeren waarbij te kiezen is uit het grote aanbod van spelers. Het goed kunnen spelen van dit spel is alleen mogelijk met een behoorlijke vaardigheid en een kennis van het spel. Dit kan verkregen worden door het kiezen van de derde optie, het PRACTICE. Kortweg vertaald betekent dit oefenen. Er zijn twee manieren om te kunnen trainen, in het spel of in het schoppen. Bij het oefenen van het spel kunnen alle onderdelen los van elkaar worden uitgeprobeerd. Hieraan zijn dan ook verder geen limieten verbonden. Via de optie CLIPBOARD kan op elk gewenst moment een overzicht worden gegeven. Hoe staat een team ervoor, wie is de leider in de competitie alles kan hier weergegeven worden. Na al deze informatie te hebben doorgewerkt is dan nu het moment gekomen om aan het spel te beginnen. Bent U er nog niet helemaal uit hoe U het spel moet spelen dan kiest U voor PRE-GAME SHOW. Nu wordt het spel door de computer zelf gespeeld, zien is leren, dat gaat zeker in dit geval op.

Tips en trucs

Het blijkt dat menigeen deze rubriek leest, gezien de vele reacties. Niet alleen tips en trucs, ook opmerkingen omtrent gepubliceerde listings en artikelen worden ons toegezonden. Om ook deze reacties een plaats te bieden hebben we besloten om deze rubriek uit te breiden tot een tips en trucs- annex lezersreactierubriek.

Heeft u nog opmerkingen (liefst geen hatelijke, dezen publiceren we toch niet!), tips en trucs of anderssoortige reacties, houdt u ze niet voor u zelf maar stuur op naar het ondertussen wel vaak genoeg genoemde adres!!!
In dit nummer weer veel routines, tips, trucs en vele andere zaken van computertonische aard. Eerst even wat anders. De problemenhoek, hier komen ze.



Problemen met C

Enige tijd geleden ontvingen we van T.M. Hoeneveld uit Enkhuizen een reactie op de C cursus. Deze cursus loopt al weer enige tijd maar de afleveringen worden met grote tussenpozen gepubliceerd. T.M. Hoeneveld had of heeft grote problemen met de listings welke gepubliceerd zijn bij deze afleveringen, in het bijzonder de aflevering welke afgedrukt stond in nr. 7, november 1988. We hebben de kleine voorbeelden weer eens ingetypt en uitgeprobeerd. Er zit inderdaad een klein foutje in. De oplossing die de schrijver aandraagt werkt echter ook niet! Wonderbaarlijk nietwaar. We hebben de brief van (waarschijnlijk de heer?!?) T.M. Hoeneveld nog eens doorgelezen. We hebben het antwoord gevonden!! Het blijkt dat onze listings, welke geschreven zijn met een Aztek C compiler, niet werken op een Lattice C compiler. We drukken hierbij de voorbeelden nog een keer af zowel voor de Aztek- als voor de Lattice compiler.

Voorbeeld 002 voor Lattice

```
main()
{
 char naam;
 printf("Wat is uw naam :");
 scanf("%s",&naam);
 printf("Hallo %s, hoe gaat
    het ?\n", &naam);
}
```

Voorbeeld 002 voor Aztek

```
main()
{
 char *naam;
 printf("Wat is uw naam :");
 scanf("%s",naam);
 printf("Hallo %s, hoe gaat
    het ?\n", naam);
```

Op zich lijken de veranderingen niet oorverdovend, doch als u de programma's laat 'runnen' zonder deze veranderingen dan zal de computer falikant op zijn 'mond' gaan!!
Compatibiliteitsproblemen zullen vaker kunnen voorkomen daar wij met de Aztek v3.6 compiler werken. Hierdoor kunnen portabiliteit problemen ontstaan met Lattice C. We hebben het vaker zien gebeuren, ook de Aztek listings kunnen niet altijd worden overgedragen op bijvoorbeeld Turbo C voor de PC. Vreemd genoeg echter treden er weer geen problemen op als we een met Aztek C geschreven programma over gaan dragen op de C compiler welke op een CYBER 962 draait. Goed we maken er geen woorden meer aan vuil. We zullen voortaan proberen de listings werkende te krijgen voor Lattice alswel Aztek.

Algemene vragen

Van Evert Pool uit Harlingen kregen we een aantal algemene vragen en een serie uiterst nuttige speltips. Hij had een vraag omtrent de Amiga machinetaal cursus. Zoals je waarschijnlijk al gezien hebt evert, zijn we opnieuw begonnen met de machinetaal cursus. Een nieuwe schrijver, een nieuwe cursus. In de vorige aflevering is het eerste deel gepubliceerd. Het volgende deel staat al op stapel dus ook dat zal niet zo lang meer duren.
Om een geassembleerd programma in Seka in te lezen dien je gebruik te maken van de 'ri' optie. Dit betekent zoveel als 'ReadIn'. Hiermee lees je een binair bestand in het geheugen. Een nadeel is wel dat je de geheugenplaats op moet geven waar je de binaire data neer wilt zetten. De gegevens komen dus op een vaste plaats in het geheugen te staan.

IFF lezer

Twee afleveringen geleden hebben we een programma gepubliceerd waarmee u IFF plaatjes, zoals die gemaakt worden met D'Paint, in kunt lezen. Er bleek echter nog een klein foutje in de compile en link instructies te zitten. Deze hebben we in Commodore INFO nr.7 gecorrigeerd. Voor de mensen die dit gemist hebben, bij het linken werd opgegeven dat u 'jiff_save.o' mee moest linken. Dit was

totaal overbodig. Deze fout was ontstaan doordat we tegelijkertijd bezig waren een 'screengrabber' te schrijven. Deze had/heeft deze file wel nodig. Reageert u nu niet meteen verbaast 'die heb ik gemist, in welke aflevering stond ie..', dit programma is nog niet afgedrukt.
Nu terug naar de reacties. M. Timmermans uit Huissen had de grootste problemen de listing 'lopende' te krijgen. Volgens hem zaten er enige fouten in de listing. We hebben het programma weer eens onder de loupe genomen, doch wij hebben geen fouten ontdekt. Naar alle waarschijnlijkheid komen uw fouten uit een heel erg bekende bron, het typefoutduiveltje. We zijn bezig met een 'checksummer' te ontwikkelen, welke deze fouten moet voorkomen.
We nemen nog contact met u op mochten er daadwerkelijke fouten in de listing zitten. In dit geval krijgt u zo snel mogelijk bericht van ons.

Amiga en uitbreidingen

Met de Amiga is het eigenlijk net weer zo gesteld als met de oude VIC20, met dit verschil dat alles op veel grotere schaal gebeurt. Ook de VIC20 kon men met vele zaken uitbreiden als men niet tevreden was met de basisconfiguratie. Bij de Amiga is het eigenlijk eender. Heeft men meer geheugen nodig dan installeert men dit. Heeft men extra achtergrondgeheugen nodig dan sluit men een extra diskdrive aan. Ook R. Euverman uit Bruchterveld kent deze problemen. Hij vroeg ons welke uitbreiding hij het best het eerst aan kon schaffen.
Menigeen zal op een gegeven moment voor dit probleem geplaatst worden. Het antwoord op de vraag, 'wat het eerst aan te schaffen' kan met een tegenvraag worden beantwoord. Wat wilt u met uw computer doen. Wat u met uw computer wilt doen is bepalend voor de configuratie van uw computer. We geven een aantal voorbeelden.

- 1. U wilt uw computer gebruiken voor D.T.P.- achtige doeleinden. Veel geheugen en een goede printer zijn dan noodzakelijk.
- 2. U wilt spelletjes spelen. Een tweede diskdrive en genoeg geheugen zijn dan plezierig.
- 3. U wilt serieus met de computer programmeren. Nu zijn een printer, een tweede diskdrive cq. harddisk en genoeg geheugen belangrijk.

Wat u dus als eerste aanschaft hangt af van wat u met de computer wilt doen.

Printeromschakel mogelijkheden

Vele Amiga 500 bezitters zijn mensen die overgestapt zijn van de Commodore 64. Meestal bezat men destijds ook een flinke stapel hardware bij deze computer. Wat zou het leven toch fijn zijn als we deze hardware ook weer op de Amiga aan konden sluiten!! Ook Michel van der Ven uit Goirle zit/zat met dit probleem. Hij bezit een Star LC-10c printer welke op de C-64 aan te sluiten is. Deze printer beschikt dus over een seriele aansluiting. Zijn vraag is nu of hij deze printer ook aan kan sluiten op de Amiga. Dit door middel van een bepaald soort interface, bijvoorbeeld de CAT & KORSH 92000g centronix interface.
Michel, naar alle waarschijnlijkheid is deze interface bedoeld om de C64 uit te rusten met een centronix interface. Het is dus niet de bedoeling om deze andersom aan te sluiten. Het kan soms wel hetzelfde effect hebben als je deze interface 'andersom' gebruikt. Toch moet je voorzichtig met dit soort grappen zijn, want een verkeerde aansluiting en je blaast een I/O chip op. Voor je computer zijn de gevolgen (niet financieel) wel te overzien, voor de printer wordt het minder. De onderdelen zijn vaak zeer specifieke chips en daarom zeer lastig te krijgen. Bovendien kosten ze 'mucho money' dus NIET DOEN. Een betere oplossing is de mogelijkheid die worden geboden door diverse bladen, bijvoorbeeld door het Duitse blad 'Amiga' van Markt en Technik, welke ombouwsets afgedrukt heeft waarmee je van je C64 een soort buffer maakt. Op deze indirecte wijze kun je toch nog gebruik maken van je C64 randapparatuur.

BOBS en sprites

Dat de Amiga fantastische grafische mogelijkheden beschikt hoef je een Amiga bezitter niet te vertellen. Het wordt een probleem echter om deze prachtige mogelijkheden in BASIC op je scherm te toveren. Ook Roland Jochems uit Hoensbroek stuitte op dit probleem. Hij wil in BASIC objecten (lees bobs) op het scherm zetten, echter hij heeft er de grootst mogelijke moeilijkheden mee om deze bobs te definieren. Zijn klacht is dat ieder die hem tot nu toe advies gegeven heeft, hem verteld heeft dat hij de 'object editor' moet gebruiken. Deze methode werkt volgens hem niet.
Goed, ook wij verwijzen je toch weer terug naar de 'object editor' daar je al over deze editor beschikt. Het is wel degelijk mogelijk om de hierin gedefinieerde bobs op het beeldscherm in BASIC te krijgen echter je moet eerst een aantal zaken afhandelen alvorens je begint met het 'displayen' van de bobs. We hebben een kort voorbeeldje geschreven om je te laten zien hoe het moet.

We hebben met de object editor een bob (=keuze 0) getekend en weggeschreven onder de naam 'test' naar drive df0:. Om dit plaatje weer in te lezen in BASIC gebruikten we het volgende programmaatje.

```
OPEN "df0:test" FOR INPUT AS
    1:REM open file
OBJECT.SHAPE
    1,INPUT$(LOF(1),1):REM
    lees 'test' in geheugen
CLOSE 1

OBJECT.X 1,10:REM set x
    positie voor bob 1 op 10
OBJECT,Y 1,50:REM set y
    positie voor bob 1 op 50

OBJECT.ON:REM laat bobs zien

FOR l=1 TO 1000:NEXT l:REM
    tel tot 1000

OBJECT.OFF:REM laat bobs weer
    verdwijnen
END
```

We hopen dat dit voldoende is om je experimenteerzucht weer aan te wakkeren.

Tot zover de vragen en antwoorden. Lets get back to the real world, natuurlijk niets minder dan het beste, hier zijn speciaal voor jullie geimporteerd uit onze archiefkasten, de 'hottest' tips en trucs!!

Waar is m'n DOS commando?

Zoals iedereen ongetwijfeld al weet heeft de nieuwe Workbench 1.3 SHELL een zeer handige 'history' buffer voor de DOS commando's. Druk op de pijltjes toetsen, UP & DOWN, en zie daar de vorige DOS commando's vliegen je om de oren. Het blijft toch wel lastig als je een specifiek commando zoekt. De enigste oplossing lijkt, aandachtig turen naar het beeldscherm en maar hopen dat je het commando niet mist. Mis! Er is een handigere oplossing. Typ de eerste paar letters in die je weet, desnoods 1 letter, en druk [SHIFT] en [pijltje_naar_boven]. Is er een commando welke overeenkomt met de opgegeven letters dan zal deze afgebeeld worden. Bestaat het commando niet, dan zal er een lege regel verschijnen.

Spelletjes of statistiek?

Veel mensen die weleens een spelletje programmeren in BASIC zullen ooit wel eens gestuit zijn op het probleem van de 'aselecte trekking zonder teruglegging', zoals het nemen van een kaart uit een kaartspel of het trekken van een nummer uit een zak met nummertjes (bingo). Het probleem zit 'm in het 'schudden'. Stel dat uit een zak met 75 nummers een voor

een de nummers worden voorgelezen. Een nummer dat al geweest is kan niet nogmaals worden getrokken, dus een random getal tussen 0 en 76 zal in toenemende mate 'al geweest' zijn. De meest simpele oplossing in Amiga-BASIC zal hiervoor al gauw 13 seconden nodig hebben. Het wordt helemaal rampzalig als je 1000 nummers moet trekken. Amiga-BASIC zorgt er dan voor dat je een noodgedwongen koffie-pauze, zeg maar gerust koffie-UUR, in moet lassen. Geen goede oplossing dus. Toch is er een andere mogelijkheid. Bekijk het onderstaande programma eens.

```
OPTION BASE 1
LIBRARY "exec.library"
DECLARE FUNCTION AllocMem&
    LIBRARY
RANDOMIZE TIMER

main:   H%=75
        DIM Array%(H%)
        BEEP: T1#=TIMER
        schudden H%

        T2#=TIMER: BEEP
        T#=T2#-T1#
        IF T#<0 THEN
    T#=T#+20864#
        IF T#<0 THEN
    T#=T#+44672#
        PRINT USING
    "##.###";T#
        LIBRARY CLOSE
        END

SUB schudden(Hoogste%) STATIC
        SHARED Array%()
        Blok%=2*(Hoogste%+1)
        Memflag&=2^0+2^1+2^16

    Adr&=AllocMem&(Blok%,Memfla
    g&)
        FOR Loop1%=0 TO
    Hoogste%-1
            POKEW
    Adr&+2*Loop1%,1+Loop1%
        NEXT Loop1%
        POKEW
    Adr&+2*Hoogste%,0
        FOR Loop2%=Hoogste%
    TO 1 STEP -1
            Q%=INT(RND*Loop2%)

    Array%(Loop2%)=PEEKW(Adr&+2
    *Q%)

    CopAant%=2*(Hoogste%-Q%)
            CopAdr&=Adr&+2*Q%
            CALL
    CopyMem(CopAdr&+2,CopAdr&,C
    opAant%)
        NEXT Loop2%
        CALL
    FreeMem(Adr&,Blok%)
END SUB
```

Het ziet er nogal rommelig uit, maar het werkt prima. Het random getal wordt gebruikt als pointer, waarna het geheugenblok na de pointer, naar 1 plaats lager als de pointer wordt gecopieerd. Hierbij wordt de inhoud van de geheugenplaats waarnaar de pointer wijst uit het geheugenblok gewist en in de array geladen. Bij de volgende trekking wordt het randommaxiumum 1 minder en de nieuwe pointer wijst het volgende nummer aan.
De snelheid-winst is indrukwekkend. AmigaBASIC deed er bij 75 getallen 0.8 seconde over. Met 1000 getallen duurde het wat langer, 11.8 seconde, maar het blijft een enorme verbetering.
Het voorlezen van de 75 getallen kan doormiddel van het volgende programmablok

```
FOR Loop1%=1 TO H%
  PRINT Array%(Loop1%)
NEXT Loop1%
```

Mocht iemand een nog snellere oplossing weten, 'please write'.
(Met dank aan R.Kohler voor de TIMER oplossingen.)

Ekke Verheul, Rotterdam.

Interceptor again

Dat nog veel mensen Interceptor spelen blijkt wel uit de volgende reactie. Als u, ondanks de door ons gepubliceerde tip, nog steeds moeite heeft met de 'qualification' van Interceptor probeer dan het volgende.

- 1. Trek direct na het opstijgen het vliegtuig op de kop zodat hij weer over het schip vliegt (hou hierbij een

```
 * Writer       : M.Timmermans
   *
 * Program      : Direk
   *
 * Abstract     : lezen van directory's
   *
 * Date         : 23-09-89
   *
 * Compile &
   *
 * linkinstr.   : cc direk.c +L
   *
 *                ln direk.o -lc32
   *

#include "stdio.h"
#include "functions.h"
#include "exec/types.h"
#include "libraries/dos.h"
#include "libraries/dosextens.h"

#define EEN 1
#define VERSION "1.00"

static struct FileLock *pdir=NULL;
static struct FileInfoBlock *dir_info;

char FBuffer[500][34],
    DBuffer[500][34];
static int FTel=EEN,DTel=EEN;


main(argc,argv)
int argc;
char *argv[];
{
 ULONG status;

 parameters(argc,argv);

 /* GEHEUGEN RESERVEREN */
 if((dir_info=AllocMem((long)sizeof(struct
   FileInfoBlock),NULL))==NULL)
   {
    printf("Not Enough memory !!\n");
    exit(FALSE);
   }
 free_pdir();     /* UNLOCK EVENTUELE OUDERE */

 /* DIRECTORY LOCKEN */

 if(!(pdir=(struct FileLock
   *)Lock(argv[EEN],(ULONG)ACCESS_READ)))
```

vervolg volgende pagina

```
} /* end main */

quit()
{
 FreeMem(dir_info, (long)sizeof(struct FileInfoBlock));
 /* geheugen vrij geven */
 free_pdir();
 exit(FALSE);
}

static free_pdir()     /* directory unlocken  */
{
 if(pdir)
   {
    UnLock(pdir);
    pdir=NULL;
   }
}

parameters(argc,argv)
int argc;
char *argv[];
{
 if(argv[EEN][NULL] == '?')
   {
    usage();
    exit(FALSE);
   }
}
usage()
{
 printf("\033[1mDirek Version %s     1989 by
    DigiSoft\033[0m\n",VERSION);
 printf("Direk laat de directory zien.\n");
 printf("Syntax: Direk <Directory>\n");
}
```

```
/* VOOR HET PRINTEN VAN
    FILENAMEN */

FPrint(FTel)
int FTel;
{
 int i;

 for(i=1;i<FTel;i++)
  printf("%s\n",FBuffer[i]);
}
```

```
/* VOOR HET PRINTEN VAN
    DIRECTORY'S */

DPrint(DTel)
int DTel;
{
 int i;

 for(i=1;i<DTel;i++)
printf("\033[33m%s\033[31m\n",
    DBuffer[i]);
}
```

```
/* SHELL SORTEER ROUTINE */

Sort(Buffer,aantal)
int aantal;
char Buffer[500][34];
{
 int afstand,i,j,L;
 afstand = (aantal/2);
 while(afstand > NULL)
  {
   for(i=(afstand+EEN);i<aan
    tal;i++)
     {
      j=i-afstand;
      while(j>NULL)
       {
        L=j+afstand;

    if(strkl(Buffer[j],Buffer[L
    ]))
         j=0;
        else
         {

    swap(Buffer[j],Buffer[L],34
    );
          j=j-afstand;
         }
      } /* end while 2 */
   } /* end for */
  afstand=(afstand/2);
 } /* end while 1 */
}
```

```
/* SWAP 2 STRINGS MET LENGTE
    L */

swap(a,b,L)
register char *a,*b;
register int L;
{
 register int i;
 register char c;

 for(i=NULL;i<L;i++)
   {
    c=*a;
    *a++=*b;
    *b++=c;
   }
}
```

```
/* COMPARE 2 STRINGS (CASE
    ONAFHANKELIJK */
int strkl(a,b)
register char *a,*b;
{
 register char ca,cb;

 do {
  ca=*a++;
  cb=*b++;
  if(ca>='A' && ca<='Z')
    ca+=32;
  if(cb>='A' && cb<='Z')
    cb+=32;
 }while(ca == cb);

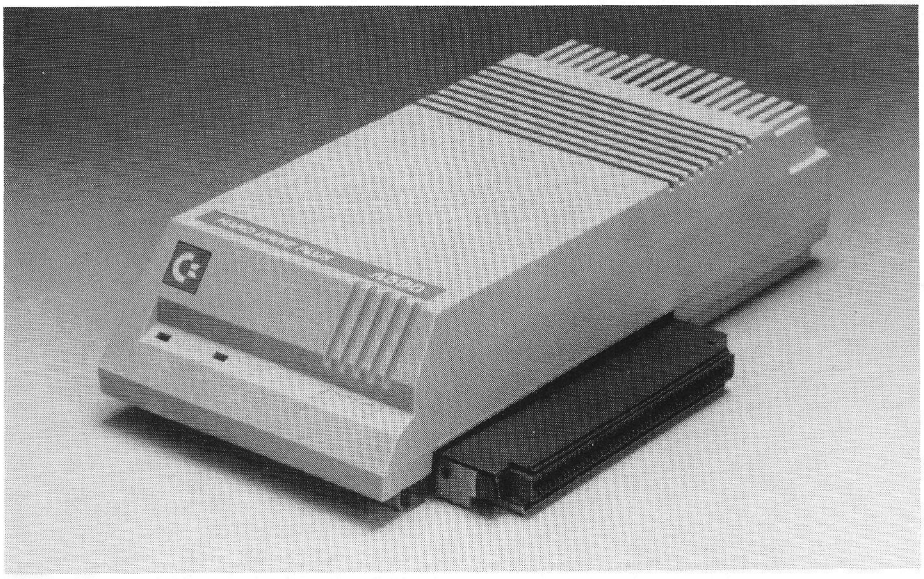
 if(ca <= cb)
   return(EEN);
 else
   return(NULL);
}
/* einde listing */
```

```
Compile en link instructies:
------------------------------
    ----------
Taal                    : C
Programma               :
    Aztek C v3.6
Compile & link instr.: cc
    direk.c +L
                        ln
```

hoogte aan van ongeveer 1000 ft.)

- 2. Druk '8' op het numerieke pad in en wacht tot de afstand tot het vliegdekschip groot genoeg is. Duw nu de joystick naar voren om weer in de normale vliegpositie te komen en om te dalen (druk nu ondertussen weer '2' op het numerieke pad in.)
- 3. Zet de motor op 50 of 60%, doe het landingsgestel en de arrestorhook uit en zet de 'brakes' aan.
- 4. Denk vooral om de hoogte. Kom niet beneden de 140 ft.

Evert Pool, Harlingen.



Efficienter filebeheer

Het laden van diskette of harddisk kan over het algemeen nog wel even sneller. Nee, geen diskaccelerator of zoiets, simpel een aanwijzing. Door de files in de volgorde van het laden naar diskette te copieren kan men behoorlijk wat tijdwinst boeken. Dus eerst de L(dir) met de belangrijkste handlers, daarna de S en de C directory etcetera. Dit levert een tijdwinst op van enkele seconden.

M.Timmermans, Huissen.

Alternatief DIR commando

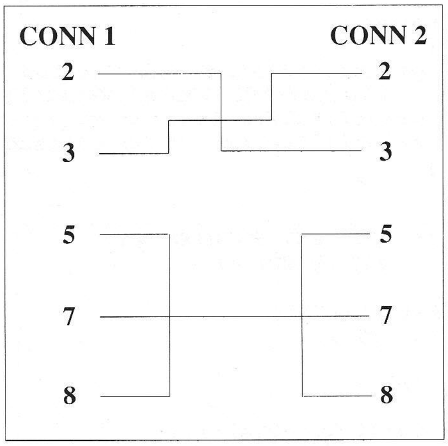
Het volgende programma leest de filenamen en de diretorynamen van disk (soft- of harddisk), sorteert deze en print deze op het beeldscherm. Als u bezig bent met een programma die de directory moet lezen, lees de onderstaande listing door en doe er uw voordeel mee.

M.Timmermans, Huissen

Amiga voor 2 personen

Heeft u wel eens met een groot mainframe gewerkt. Was u er ook even verwonderd over dat een computer enkele honderden gebruikers kon bedienen? Ja? Dan is de volgende tip iets voor u.
Met Workbench 1.3 is er een nieuwe gebruiksmogelijkheid van uw Amiga bij gekomen. Het is mogelijk om met z'n tweeën aan een Amiga te werken. Het is alleen wel vereist dat u nog over een tweede computer beschikt. Deze tweede computer wordt nu ingezet als een terminal. Voor deze computer dient u ook te beschikken over een terminal programma. Heeft u deze niet dan kunt u altijd een klein communicatieprogrammaatje in bijvoorbeeld BASIC schrijven. Dit is vaak wel afdoende.
Wat moet u instellen op uw Amiga. Als eerste moeten de RS232-parameters, dus de instellingen van de seriele poort, ingesteld worden.Dit doen we met behulp van de Preferences. Kies voor 'change serial'. Beschikt u nog over een C64 of C128 dan kan de baudrate op z'n hoogst 2400 BAUD zijn. Alle andere instellingen, zoals parity, stopbits, etcetera zijn afhankelijk van het gebruikte terminalprogramma aan de andere zijde. Weet u het niet, zet parity op none, de stopbits op 1, de read bits en de send bits op 8 en de handshaking op none. De buffersize zal waarschijnlijk op 4096 staan. Laat deze maar zo staan. Verlaat dit menu nu weer en save de instellingen.
De bekabeling tussen de twee computers. U dient twee 25 pens female D connectoren aan te schaffen. Voor de C64 of C128 is een andere oplossing mogelijk. Wilt u hier meer over weten, schrijf dan naar de redactie. Voor de verbinding hebben we besloten een 3 lijns verbinding te nemen. Hierbij gebruiken we alleen de transmit, receive en de groundsignalen. Sluit pen 2 van connector 1 op pen 3 van connector 2 aan en pen 3 van connector 1 op pen 2 van connector 2. Koppel vervolgens de twee grounds van de twee pluggen aan elkaar, dus pen 7 van connector 1 en pen 7 van connector 2. Omdat sommige terminalprogramma's nogal eens eigenwijs zijn verbinden we aan beide connectoren ook pen 5 aan pen 8 door. Dit dient u dus op BEIDE connectoren te doen. Kijk voor de zekerheid onderstaand schema door of u het wel correct gedaan heeft.
Nu verbinden we de twee computers met elkaar. Denk erom, ZET BEIDE COMPUTERS EERST UIT!!! Duw de pluggen in de seriele connector van de computers. Zet de computers nu weer aan. Start ze op. Start op de computer die als terminal gaat dienen een terminalprogramma op. Dat is alles voor de terminalzijde. De Amiga weer. U bent nu weer terug in de SHELL. Typ hierin in, **newshell aux:**. Als er nu een requester verschijnt met de tekst 'insert volume aux:' dan zult u eerst het device aux: moeten 'mounten'. Kies bij het requester voor de optie 'cancel', en typ vervolgens in 'mount aux:'. Typ hierna weer in **newshell aux:**. Als alles goed is ziet u nu op de terminalcomputer een opstartmelding verschijnen. Ziet u onleesbare tekens dan staan de seriele poort parameters niet goed ingesteld. Deze kunt u in het terminal programma wel veranderen. Als u eenmaal 'verbinding' heeft kunt op de terminal in AmigaDOS werken.

Afbeelding 1 NULL Modem kabel.



Opmerking bij afbeelding 1. Een aantal draden kruisen in dit schema. Dit betekent NIET DAT ZE DOORVERBONDEN moeten worden.

Tot slot

Tot zover deze aflevering van onze 'ever continuing tips & trucs story, starring ....
Johan F as: Johan 'tips' Carrington.

Johan v.R as: Johan 'genius' Colby.

Computer assistance:
Commodore Amiga 2000 & 500

Writers assistance: WordPerfect Corp.

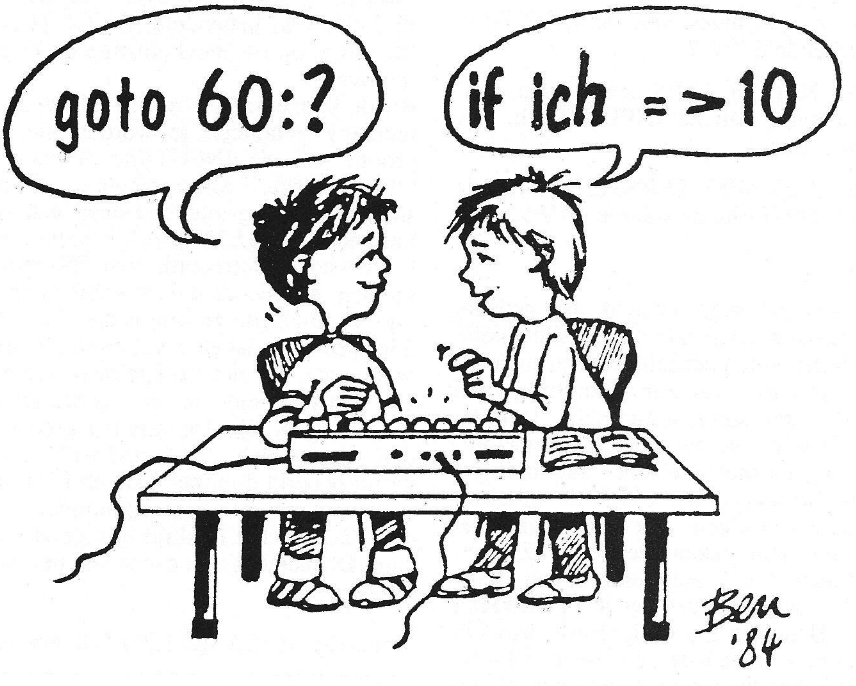
PS/ Thanks also to my dog, for not eating my floppies.

Tot de volgende keer.

Johan & johan.

On-beveiligen van Basic

Daar zit je dan met je voor ƒ 15,- gekochte programma (spel). Er zijn 30 levels mogelijk, maar middenin level-17 (meestal in dezelfde spelsituatie) gaat het fout (altijd). FLITS, wit scherm, dat is het einde. Een of andere leukerd heeft alle kleuren op wit gezet, waardoor iedere melding onzichtbaar wordt en blijft.



Lichtpuntje

Gelukkig is het programma geschreven in AmigaBasic. Even LISTen en op onderzoek uitgaan. FOUT, er valt niets te LISTen, het programma is protected (geSAVEd met optie-P). Op het scherm komt netjes de melding : ILLEGAL FUNCTION CALL. Tja, en toen ? Het AmigaBasic manual meldt op blz. 8-129 "... any attempt to list or edit it will fail". Hier en daar eens geïnformeerd. Een aantal oudere tijdschriften eens beter bestudeerd, maar geen enkel lichtpuntje.
Dan maar zelf in de aanval. Een poging doen om het geheim van het protecten te ontdekken. Er zijn hierbij TWEE onderdelen: **EEN** --- Ontdek het geheim van het protecten, **TWEE** --- maak het protected-zijn ongedaan. Punt EEN is uiteraard het allerbelangrijkste. En: trek je niets aan van mensen, die ACHTERAF weten te melden, dat het eigenlijk erg logisch en eenvoudig is.

Het protect-geheim

Maar goed. Een klein programma samenstellen. Dit programma B-SAVEn en P-SAVEn (SAVEn met optie-B en met optie-P) en de inhoud van beide aldus ontstane files hexadecimaal printen en bestuderen. Na dit een aantal keren (steeds met een iets ander programma) gedaan te hebben, is mij opgevallen dat de *Eerste byte* van een **B-SAVE** programma altijd &HF5 of wel CHR$(245) is en dat de *Eerste byte* van een **P-SAVE** programma altijd &HF4 of wel CHR$(244) is. Veel meer schoot ik er niet mee op.
Toen heb ik vier "programma's" samengesteld, elk bestaande uit ongeveer 400 dezelfde characters. EENmaal 400 "A", EENmaal 400 "B", EENmaal 400 "9", EENmaal 400 "=". Alle vier programma's B-SAVEn en P-SAVEn en de aldus ontstane 8 files hexadecimaal printen en bestuderen. Tijdens deze studie heb ik de "grap" ontdekt. Deze "grap" zal ik nu trachten te omschrijven.

Ik moet nu eerst een aantal begrippen introduceren.

```
ABF= Aantal Bytes in de File  (lengte
van programma).
RNG= RaNGnummer van een byte in de
File (1 t/m ABF).
POS = POSitie van een byte in de File
(1 t/m 143).
B-byte = byte van programma, geSAVEd
met optie-B (&H00 t/m &HFF)
P-byte = byte van programma, geSAVEd
met optie-P (&H00 t/m &HFF)
```

Elke byte van een programma (= file) krijgt **in gedachten** een RaNGnummer en een POSitie toegewezen. Dit gaat als volgt: voor de eerste byte geldt: RNG = 1 en POS = 1. Voor de volgende bytes geldt: " RNG = RNG + 1 " en " POS = POS + 1 : if POS 143 then POS = 1 ". Gevolg: elke RNG komt slechts EENmaal voor, elke POS kan meerdere keren voorkomen. Bv. de bytes met RNG=100, met RNG=243, met RNG=386 en met RNG=529 hebben alle vier POS=100.
Verder moeten we in gedachten een tweedimensionale tabel voorstellen, waarbij de ene Dimensie wordt gevormd door de waarde van een B-byte en de andere Dimensie wordt gevormd door de POSitie van een byte, terwijl elk element gevuld wordt met de waarde van een P-byte. Dit betekent een tabel van 256 x 143 = 36.608 elementen van 1 character.

Deze CONVERSIE-tabel heeft dus de volgende 3 kenmerken:
"waarde B-byte", "POSitie byte", "waarde P-byte".

Als er 2 bekend zijn, dan kan de derde bepaald worden. Van een byte is de POSitie ALTIJD bekend. M.a.w. als we de waarde van een B-byte weten, kunnen we de waarde van de bijbehorende P-byte bepalen EN ... omgekeerd.
Tijdens het protecten door het programma "AmigaBasic" zijn de "POS byte" en "waarde B-byte" bekend en wordt in de tabel de "waarde P-byte" opgezocht. "AmigaBasic" voert de opdracht (SAVE "progname",P) razendsnel uit. Zo snel, dat je er eigenlijk niets van merkt. Mogelijk handelt "AmigaBasic" iets anders, dan ik zojuist heb omschreven, maar het effect is hetzelfde.
Tijdens het UNprotecten door mijn programma "Unp" zijn de "POS byte" en de "waarde P-byte" bekend en wordt in de tabel de "waarde B-byte" opgezocht. "Unp" werkt veel en veel langzamer dan "AmigaBasic". Het werkt wel volgens het omschreven principe.

Principe

Principe van "Unp" (wie het precies wil weten: zie LISTING):

° 1. LEES een aantal bytes van de IN-

PUT-file.

- 2. CONVERTEER elke P-byte m.b.v. CONVERSIE-tabel naar een B-byte
- 3. SCHRIJF deze B-bytes naar de OUTPUT-file.
- 4. ALLE bytes van de INPUT-file behandeld ?????
- A. NEE: a. LEES een volgend aantal bytes van de INPUT-file b. GA naar stap 2.
- B. JA: GEEF EERSTE byte van de OUTPUT-file de waarde &HF5.

UNP

Dit is eenvoudiger gezegd, dan gedaan. Allereerst moet de inhoud van alle 36.608 tabel-elementen bepaald worden en de tabel moet foutloos zijn. Een tijdrovend karwei, dat soms onmogelijk leek en waarmee ik u verder niet zal vermoeien. Vervolgens moet de tabel uiteraard gebruikt worden door "Unp". De tabel op diskette gebruiken, zou "Unp" wel heel erg langzaam maken. De tabel in zijn geheel door "Unp" laten inlezen, kan alleen als je veel geheugen tot je beschikking hebt. Bedenk, dat AmigaBasic aan elk element 4 extra bytes toevoegt. Elk element wordt dus 5 bytes groot i.p.v. 1 byte. De CONVERSIE-tabel is 256 x 143 = 36.608 bytes groot. Deze tabel zal dus in "Unp" 36.608 x 5 = 183.040 bytes = 179k in beslag nemen. Een bezitter van een Amiga met 512k kan in AmigaBasic niet voldoende ruimte creeren om daar "Unp" compleet met tabel in te zetten. Ruimte creeren gebeurt m.b.v. het AmigaBasic-commando: CLEAR (zie AmigaBasic-manual blz 8-33). Uiteindelijk heb ik in "Unp" een tabel gedefinieerd van 143 elementen van 256 characters lang en dat kost "slechts" 143 x (256 + 5) = 37.323 bytes of wel ca. 37k. Nu kan elke Amiga-gebruiker het programma "Unp" draaien , nadat het AmigaBasic-commando (CLEAR ,60000) is gegeven. Extra voorwaarde is wel, dat er verder zo min mogelijk geheugen in beslag genomen wordt.
Note: Het verhaal over het CLEAR-commando is alleen van toepassing voor de niet-gecompileerde versie.

Input-output files

Een eveneens belangrijk punt is, dat de OUTPUT-file precies evenlang moet worden, als de INPUT-file. Anders heb je later kans op een INTERNAL ERROR als je de OUTPUT-file aan het EDITen bent in AmigaBasic. De INPUT-file wordt RANDOM gelezen en de OUTPUT-file wordt RANDOM geschreven. De keuze van recordlengte=1 geeft op de meest eenvoudige manier het gewenste resultaat, maar is wel erg langzaam. Ik heb uiteindelijk gekozen voor een mix van recordlengte=256 en recordlengte=1. Van de INPUT-file worden net zo lang records van 256 bytes gelezen totdat er minder dan 256 bytes over zijn.. Daarna worden er records van 1 byte gelezen (max 255) totdat alle bytes van de INPUT-file zijn behandeld. De OUTPUT-file wordt op overeenkomstige wijze geschreven.
Bij de keuze van de record-tellers dient rekening gehouden te worden met de grootte van de INPUT-file en van de OUTPUT-file. Short-integers werken sneller dan long-integers, maar een file kan meer dan 32.767 bytes bevatten. Bij het eerste deel (records van 256-bytes) kunnen de record-tellers short-integers zijn. De max file-grootte is dan 32.767 * 256 = ruim 9 diskettes vol en dat is ruim voldoende. Bij het tweede deel (records van 1-byte) moeten de record-tellers long-integers zijn. De max file-grootte is dan theoretisch 2.147.483.647, maar wordt beperkt door het GET en PUT statement (max recordnummer = 16.777.215). Dit is altijd nog goed voor ruim 18 diskettes vol, eveneens ruim voldoende.

Note: Bij AC/BASIC 1.2 heb ik ervaren, dat het GET en PUT statement een max recordnummer = 32.767 hebben. Uiterst vervelend. Daar heb ik wel iets op gevonden, hoewel ik er zelf niet gelukkig mee ben (zie in listing Unp.acb).

Schatting

Als laatste nog een opmerking over de "schatting". De eerste versies van het programma Unp waren uitermate traag (doorlooptijd bijna 2 uur voor een file van ruim 13.000 bytes). Toen heb ik de subroutine "schatting" geschreven, die een redelijke schatting geeft van de te verwachten RUN-tijd. Deze schatting wordt gedaan nadat ca 10% van de INPUT-bytes is behandeld. De nauwkeurigheid van de schatting is sterk afhankelijk van de gekozen opslag-media (df0: ram: hd0:) en van de overige taken, die de Amiga van u heeft opgedragen gekregen.

R.C.A.B. Kohler
Gerrit van der Veenstraat 37
3762 XH Soest

Public Domain

Van de listings en de daarbij behorende conversie-tabel, is een public domain schijf gemaakt, die voor *f* 12,50 (incl. BTW en verzendkosten) te bestellen is bij Infolist, giro 3157656 t.n.v. Infolist Amsterdam, onder vermelding van Unprotect disk. Voor België: stort Bfr. 300 op BBL nr. 310050602562 t.n.v. SAC.
Deze schijf bevat vier bestanden:

- **Read.Me** bevat belangrijke informatie, waarvan ik vind dat men deze gelezen moet hebben alvorens men met het programma Unp van start gaat. Deze file is gecreeerd met ED.
- **Unp** is het met AC/BASIC gecompileerde AmigaBasic programma, dat m.b.v. de file Unprot.d143 (File-7) het UNPROTECTEN kan verzorgen.
- **Unprot.d143** is de CONVERSIE-tabel, die onontbeerlijk is voor het programma Unp (File-6).
- **Unp.acb** bevat de source, geSAVEd met optie-A, die als input heeft gediend voor de AC/BASIC-compiler. Deze compiler heeft o.a. de file Unp.acb.run (de PHASE) gecreerd, welke ik heb geRENAMEd tot Unp. In deze source definieer ik een eigen SCREEN en WINDOW om te voorkomen, dat na elke ENTER het scherm wordt opgebouwd, schoongemaakt en weer opgebouwd.
- **Unp.bas** bevat source, geSAVEd met optie-B, die vanuit AmigaBasic geRUNned kan worden. In deze source gebruik ik het standaard output-scherm. Alvorens het programma te RUNnen, dient u precies EEN keer (niet vaker) het AmigaBasic commando (CLEAR ,60000) te hebben gegeven. Het programma vraagt er ook naar.

De listings publiceren bleek niet zo zinvol, aangezien het programma alleen werkt als de CONVERSIE-tabel beschikbaar is. Deze file is 36.608 bytes groot en moet hexadecimaal worden geprint, waarbij voor de leesbaarheid een blank tussen 2 opeenvolgende bytes moet worden gezet. Dit betekent ca. 3 x 36.608 = 109.824 characters. Bij 100 regels van 100 characters betekent dit toch 11 blz. vol. En dan staat het nog niet in de juiste vorm op diskette.

VES ONE

Complete geïntegreerde videostudio voor de Amiga

Professionele Videobewerking met de Commodore Amiga bestaat uit drie componenten: titels genereren, het aanbrengen van speciale effecten en beeld- & kleurmanipulaties. Dikwijls zijn daar gescheiden hardware- en softwarecomponenten voor nodig. Echter niet bij de geïntegreerde VES ONE van Videocomp, die een alles inéén oplossing biedt.

De bekende Duitse producent van videorandapparatuur Videocomp kan het niet laten elk half jaar weer een nieuw baanbrekend Amiga-produkt uit te brengen. Ook nu hebben deze voortvarende Frankfurters waarschijnlijk weer in de desktopvideoroos geschoten. Het geïntegreerde VES ONE (= Video Effect Systeem) biedt alles wat de gevorderde videohobbyist zich maar kan wensen: veel mogelijkheden, compacte alles inéén bouw en eenvoudige bediening.

Het is verbazingwekkend hoeveel videoprodukten voor de Amiga de laatste tijd op de markt gekomen zijn. Blijkbaar wil de gevorderde videohobbyist meer dan alleen maar wat losse scènes aan elkaar plakken. Het is allemaal speciale effecten, professionele aftiteling en geavanceerde beeldmanipulatie wat de klok slaat. En vlak natuurlijk de no- en low-budget videoproducenten niet uit. Die krijgen met de Commodore Amiga en produkten zoals de VES ONE echte professionele mogelijkheden is huis.





Komplettlösung

Videocomp zelf noemt VES ONE met zo'n beeldsprakige duitse term de 'Komplettlösung'. Natuurlijk wel een wat te prentieuze claim. Niettemin is dit Video Effect Systeem één van de modernste en veelzijdigste studio-units die in deze prijsklasse voor de Commodore Amiga 500 en 2000 verkrijgbaar is. Laten we de verschillende onderdelen eens stuk voor stuk bekijken.

Titelgenerator

Titelgeneratoren vindt men tegenwoordig als losse units of ingebouwd in camrecorder, videocassetterecorders (VCR's) en apparatuur voor videobewerking (Special Effect Generators, [SEG's] en videoprocessoren). De in camrecorders of VCR's ingebouwde titelgeneratoren zijn doorgaans beperkt van aard. Slechts enkele lettertypen, kleuren en soms wat bewegingsmogelijkheden staan ter beschikking. De Amiga biedt daarentegen aanzienlijk meer:

- Een groot aantal fonts (= verschillende lettertypen en vormen)
- 4096 kleuren voor de letters of achtergrond. Tevens verschillende graderingen voor verlopende achtergronden e.d.
- Vloeiende bewegingen, het zogenaamde scrollen van links naar rechts, boven naar beneden en omgekeerd of via een schuintraject
- Vormveranderingen van de letters. Bijvoorbeeld in grootte, lengte/breedte of in/uitzoomen

Bij VES ONE drijven de titeleffecten grotendeels op de meegeleverde **Videopage** software. De mogelijkheden zijn één typeface met vier verschillende fonts, 8 verschillende kleuren, scrollen/lopen en 16 in/out fade- respectievelijk superimpose-effecten per titelpagina. Meer zal de kleine videoproducent niet nodig hebben. De VES ONE hardware zorgt voor de **genlock** faciliteiten om het RGB-computersignaal met het gewone VHS of Super-VHS videosignaal te combineren.

Videodigitizer

Voor het overzetten van videobeelden in een grafisch Amiga IFF-bestand is een **vi-**

deodigitizer nodig. Videocomp greep hiervoor naar de beproefde DIGI-VIEW-GOLD digitizer van NewTek. Deze videodigitizer is volledig in het VES ONE systeem geïntegreerd. DIGI-VIEW-GOLD biedt onder andere de volgende mogelijkheden:

- Een oplossend vermogen van 320 x 256 tot 768 x 584 beeldpunten (pixels)
- 16 grijstrappen
- 2 tot maximaal 4096 kleuren
- Beeldmanipulatie; kleur, contrast, helderheid, beeldscherpte, kleurpalet en verschillende schermresoluties (= aantal pixels)

De besturing vindt plaats vanaf het VES ONE voorfront (vijf draaiknoppen en één drukknop) en de DIGI-VIEW-GOLD software. Verder kunt u de gedigitaliseerde beelden altijd met andere tekensoftware bewerken.

Videomixer

Centraal in het VES ONE-systeem staat de **geïntegreerde videomixer.** Deze mixer mengt de verschillende Amiga- en videosignalen tot (naar keuze) een gewone of Super-VHS output naar de montage-VCR. Daarbij heeft de gebruiker de keuze uit:

- Drie verschillende wipes
- Superimpose
- Fade in/out

Alle effecten zijn vanaf het voorfront van de VES ONE unit regelbaar. Het aanbod aan wipes is kleiner dan bij de doorsnee losse SEG-unit zoals bijvoorbeeld de JVC S-VHS videoprocessor JX SV 77.

Signaalprocessor

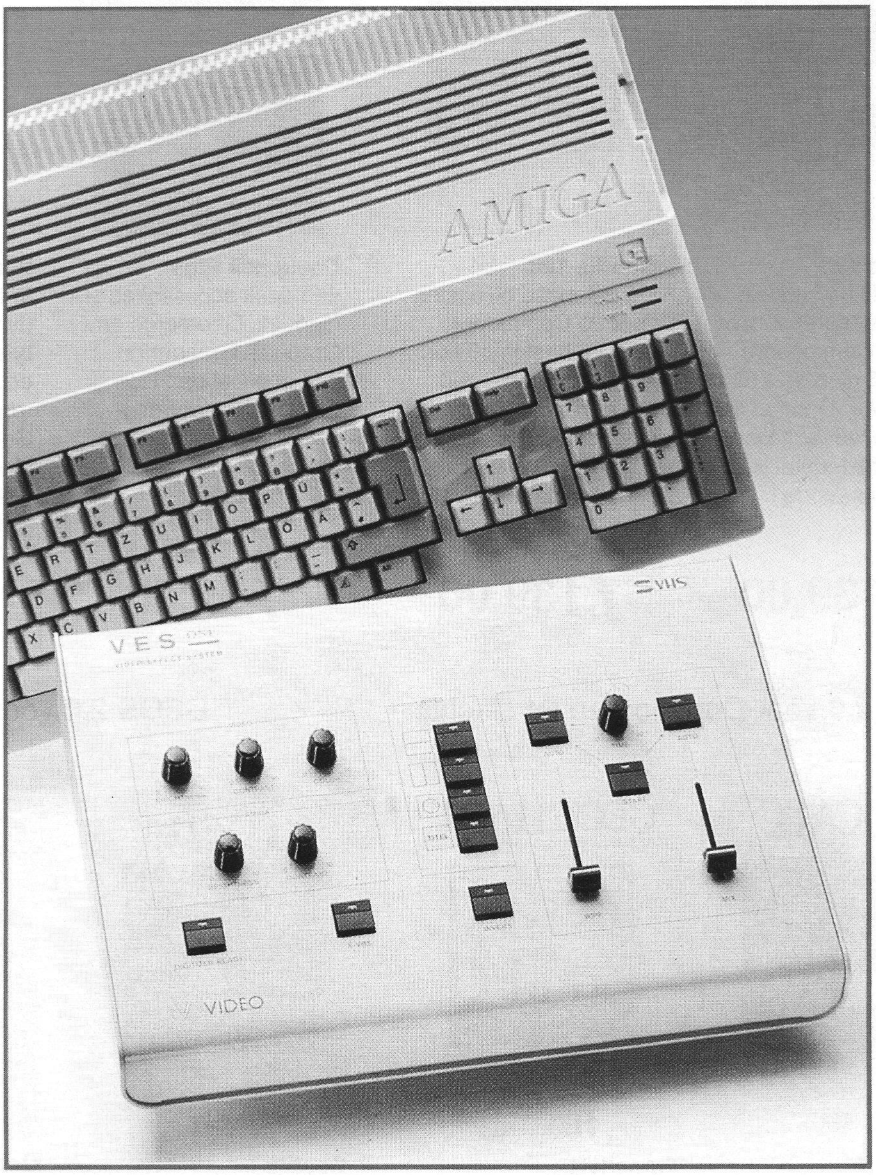
Videosignalen zijn gecompliceerde gegevensstromen. De kans is groot dat er bij de vele tussentijdse studiobewerkingen iets mee misgaat en dat geeft kwaliteitsverlies, kleurveranderingen of storende ruis. Moderne studio-units zoals de hier besproken VES ONE bevatten daarom een **videosignaalprocessor** die de overspeelverliezen zoveel mogelijk probeert terug te brengen. In combinatie met de Commodore Amiga worden contrast, helderheid en kleurverzadiging door de signaalprocessoren bewaakt.

De unit

De VES ONE zelf bestaat uit een compact (340 x 240 x 70 mm) kastje met afgeronde hoeken. Aan de voorzijde zit het bedieningspaneel. Daarop zitten draaiknoppen voor de regeling van het videosignaal (helderheid, contrast en kleur), het Amiga- signaal (helderheid en kontrast), drukknoppen voor de digitizer, S-VHS, reverse, de drie wipes en titel-superimpose.

De tv-norm is CCIR/PAL en de bandbreedte bedraagt 5,5 MHz.

De aan het begin van dit artikel genoemde term Komplettlösung wordt ons inziens



Het tijdgedeelte voor de wipes en videomix wordt geregeld door een draaiknop, startknop en twee auto-knoppen. De wipe en mix kunnen ook handmatig met de schuifregelaars bediend worden. Aan de achterzijde zitten de volgende aansluitingen:

- Een eigen drie-aderige netvoeding
- De Amiga-connectors voor de RGB monitor, parallel-, game- en RGB-poort
- Gewoon VHS; BNC video-out (2) en video-in (1)
- Super-VHS; Ronde camrecorder-plug video-out (2) en S-VHS-in (1)

op het echt professionele vlak niet geheel gehaald. Daar is de VES ONE dan ook niet voor bedoeld. In de doelgroep serieuze hobbyist en low- tot no-budget videoproducent zal deze studio zijn weg zeker wel vinden. Compactheid en relatief veel mogelijkheden voor een redelijke prijs zullen deze categorie gebruikers best aanspreken. De echte professional zal naar andere apparatuur (overigens ook door Videocomp geleverd) moeten uitkijken en vooral veel geld dienen mee te brengen. Voor informatie:
Catronix, 010 - 4507696
Videocomp, Berner Strasse 17
6000 Frankfurt/M 56, West-Duitsland

U.S.

AMI...Alignment System

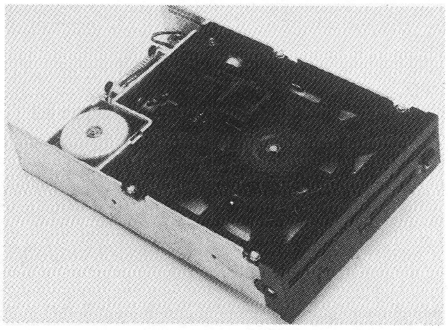
Heeft u vaak last van read-write errors of anderssoortige diskdrive fouten, dan lijkt het er veel op, dat de lees/schrijf kop van uw diskdrive niet meer goed staat afgesteld. De oplossing was, breng uw Amiga of de drive naar de reparateur. Aan deze oplossing hing altijd een aardig prijskaartje (natuurlijk ten voordele van de reparateur). Komen dergelijke diskdrive problemen u bekend voor dan hebben we hier de oplossing voor u. AMI...ALIGNMENT SYSTEM!

AMI...ALIGNMENT SYSTEM

Het is zoals de fabrikant zegt, 'a precision floppy drive head alignment and performance monitoring system', ofwel vertaald naar gewoon Nederlands 'een programma om de lees-schrijf kop van de diskdrive af te stellen of om te kijken hoe goed het gesteld is met de lees-schrijfkoppen van uw drive'.

Het leukste, het uitpakken

Bij het uitpakken valt ons een ding meteen op, de karige handleiding. Onze versie was uitgerust met een, klein uitgevallen, aantal snel ingebonden A4-tjes. We houden het er maar op dat dit te wijten is aan het feit dat het hier om een demoversie gaat. Het AMI...ALIGNMENT SYSTEM wordt geleverd met twee diskettes waarvan de ene diskette fungeert als bescherming tegen illegaal kopiëren. De tweede diskette, de zogenaamde 'calibratie'-diskette, valt bijna niet te kopiëren. Althans, als deze gekopieerd wordt dan is het programma eigenlijk waardeloos daar met deze calibratie-diskette de kwaliteit van de diskdrive bepaalt wordt. Wordt deze nu gecopieerd dan zal deze, nooit helemaal goede drive meteen zijn sporen achterlaten wat de meetresultaten negatief zal beinvloeden.
De andere diskette is de eigenlijke 'program' diskette. Op deze diskette staan een aantal files waarvan de belangrijkste 'ALIGN' heet. Start het programma op door op dit icon te clicken of door de computer te resetten en de diskette in de drive te doen.



Het programma Align

Het programma start op met een melding van de producent omtrent de rechten. Het eigenlijke programma wordt pas zichtbaar nadat op de spatiebalk gedrukt is. In eerste instantie zal men denken dat het hier om een spelletje gaat, doch het gebrek aan snel flitsende graphics en bijbehorende 'samples' overtuigen ons dan toch van het feit dat we hier met een echt 'integer' programma van doen hebben.
Wat zien we. Een, zoals de producent beweert, 'space-ship style control-panel'. Wat meteen opvalt is het beeldscherm in de rechterbovenhoek welke dient om systeem mededelingen door te geven. Verder zijn nog een aantal 'knoppen' en 'lampjes' zichtbaar. Door een of meer van deze 'knopjes' te selecteren, krijgt men toegang tot de verschillende testfases die het programma herbergt.

De verschillende controles

Er zijn vier verschillende tests met het programma mogelijk. Dit zijn:

- 1. The Automatic alignment test;
- 2. Drive motor speed test;
- 3. The Performance test;
- 4. The Manual Drive test.

Alle vier dienen ze om een bepaald aspect van de diskdrive te testen.
We nemen de vier testen afzonderlijk door.

De automatic alignment test.

Wilt u een snel overzicht hebben van de werking van uw diskdrive kies dan voor deze test. Deze test scant de disk op drie verschillende locaties, te weten op track 0, track 39 en track 79, en bepaalt aldaar de correctheid van de werking van uw diskdrive. Als alles goed gaat zal de melding 'Satisfactory' op het scherm verschijnen. U hoeft dan niets te corrigeren aan uw drive. Als de melding 'Unsatisfactory' een keer verschijnt tijdens de test dan is er iets mis met de diskdrive. U zult uw diskdrive moeten corrigeren. Ook hierop heeft de producent gerekent. Hierover zometeen meer.

De drive motor speed test.

Deze test bepaalt aan de hand van de 'index sinc mark' het aantal omlopen per minuut. Uit deze gegevens kunt u een aantal zaken afleiden, zoals de conditie van de drive motor, de conditie van voedingsspanning, de algemene mechanische conditie van de drive en de conditie van de testdiskette. Door de resultaten te vergelijken met wat men weet (een Amiga drive hoort ongeveer 300 omlopen per minuut te draaien) kunt u een conclusie trekken omtrent de conditie van de drive.

De performance test.

Dit is de enige echt verwoestende test. Ze schrijft data op de disk waardoor deze daarna eerst weer opnieuw moet worden geformatteerd om gebruikt te kunnen worden.
De performance test kent drie verschillende subtests. Ten eerste is er de 'track integrity test'. Deze bepaalt in hoeverre er sprake is van een goed medium, dus de kwaliteit van de gebruikte diskette. Bij een goede track zal de melding 'VALID TRACK' verschijnen. Gebeurt dit niet, tja, gooi de schijf dan maar in de vuilnisbak (waardeloze white labels ook). De tweede subtest is de 'write test'. Hiermee kan worden bepaald of de lees-schrijf kop smerig is. Op het beeldscherm verschijnt het aantal bytes dat geschreven is naar disk. Hieruit wordt een gemiddelde bepaald. Als het gemiddelde kleiner is dan de constante waarde, pak de alcohol maar en een wattenstaafje, nee ho stop! Maak de schade niet groter dan deze al is. Lees de handleiding eerst maar goed! De derde subtest bepaalt het aantal gelezen bytes per seconde. Deze subtest heet dan ook 'read test'. Ook voor deze test geldt in grote lijnen weer hetzelfde als voor de vorige. Ze leest bytes van schijf en bepaalt hiervan het gemiddelde. Als deze waarde

onder de constante waarde ligt, is de kop smerig.
Aan het einde van deze drie subtests wordt het gemiddelde op het beeldscherm afgebeeld. Hieruit kunt u nu uw conclusies trekken.

De vierde en laatste test.

Dit is de manual mode drive control test. Als u de voorgaande testen met de hand uit wilt voeren, kiest u voor deze test. Hiermee kunt u 'manueel', dus alles zelf doen en de instellingen bewerken: de drive motor aan of uit zetten, andere tracks instellen en tracks testen. Van deze acties krijgt u weer de resultaten op het beeldscherm te zien. Deze test is vooral gemakkelijk als u bezig bent de read-write kop bij te stellen. Dit bijstellen van de kop wordt goed uitgelegd in het handboekje.

Align en de printer

Alle resultaten kunt u ook naar de printer sturen. Click op het 'knopje' print en de resultaten zullen naar de printer worden gestuurd. Alle vier tests sturen hun gegevens dan naar de printer. Zie afbeelding 1. Elke test begint met wat header informatie. Er wordt vermeld welke test in werking is, welke drive getest wordt en de huidige datum en tijd worden vermeld.

```
*************************************************************
                 AMI...ALIGNMENT REPORT for Drive: DF0:
AUTO ALIGNMENT TEST
Thu Nov 16 20:13:27 1989
*************************************************************
TARGET TRK      ACTUAL TRK        TIME       ALIGNMENT CONDITION
-------------------------------------------------------------
00              00                  3%         SATISFACTORY
39              39                 65%         SATISFACTORY
79              79                 68%         SATISFACTORY
Completed auto pass. General condition:      SATISFACTORY

00              00                 65%         SATISFACTORY
39              39                 68%         SATISFACTORY
79              79                 68%         SATISFACTORY
Completed auto pass. General condition:      SATISFACTORY

End auto alignment test
```

Afbeelding 1. Uitdraai op de printer van 'auto alignment test'.

Wat bij een slechte drive?

Als u nu onverhoopt toch de melding op het beeldscherm of printer mocht krijgen dat uw diskdrive(s) 'misaligned' zijn, raak dan niet in paniek. De handleiding beschrijft uitgebreid hoe u de koppen weer bij kunt stellen. Voor een aantal soorten drives wordt een beschrijving gegeven waar u de veranderingen aan moet brengen. Er worden vijf drives behandeld dus meer dan genoeg kans dat uw type er bij zit. Denk er echter wel om, als u de drives uit elkaar haalt, u daarmee wel eventuele garantie verliest!!

Conclusies

AMI...ALIGNMENT SYSTEM is een zeer nuttig instrument als u beschikt over niet goed functionerende drives. Alles wat u nodig heeft voor het goed afstellen van de drives, is in dit pakket aanwezig. Een punt van kritiek is echter wel de zeer karige handleiding.
Het geheel komt in een zeer kleurig omhulsel, waarvan de inhoud bestaat uit twee diskettes en een handleiding.

Johan & johan

Amiga-towers

Kracht vanuit de "hoogte"

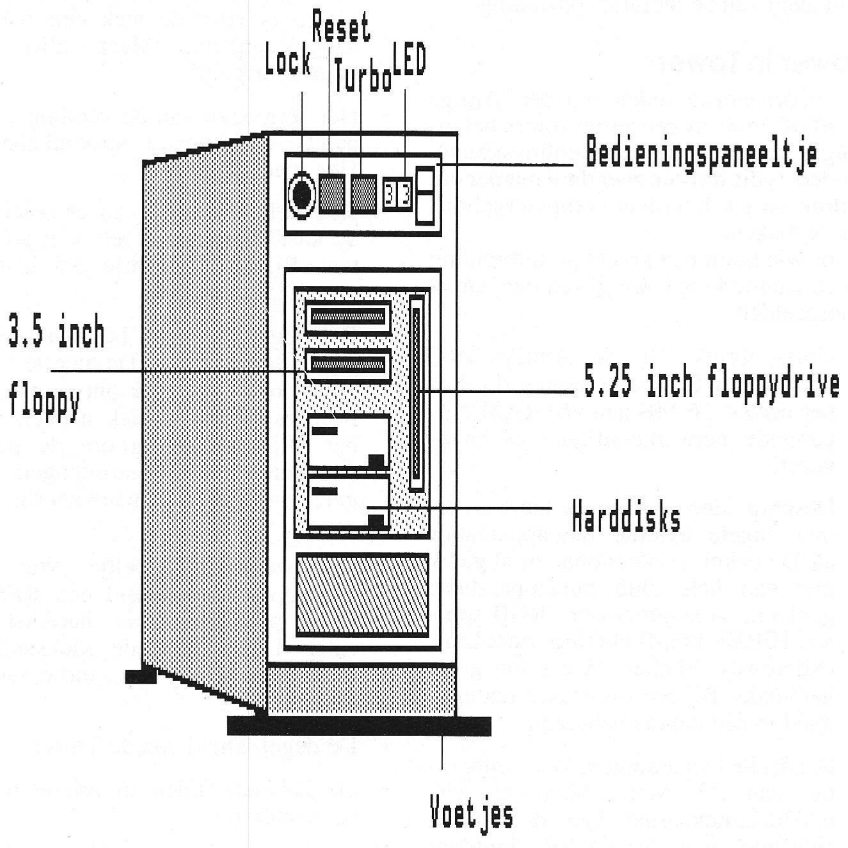
**Bij de IBM compatibele PC is de tower-systeemkast een bekend verschijnsel. De torens zijn er in grote vloermaten en kleinere bureau-uitvoeringen.
Met de recente professionele pretenties van de Amiga 2000 en 2500 wordt de torenkast ook voor deze Commodore-telgen interessant. Veel kracht voor veel geld lijkt het hierbij het devies.**

Een volle systeemkast

Aan de Amiga 2000/2500-lijn worden steeds hogere eisen gesteld. Meer RAM-geheugen, steeds snellere turbo-kaarten, grotere snellere harddisks, inbouw van tapestreamers en CD-ROMS, een extra videokaart of genlock, noem maar op. De postorderbedrijven kunnen het u leveren. Bij de Amiga 500 is daarvoor geen ruimte en groeit het bureau al gauw vol met externe randapparatuur. De systeemkast van de Amiga 2000/2500 is van het grote AT-kastformaat kan wel wat extra's aan. Ook hier zijn de mogelijkheden echter begrenst. Met drie drives (bijvoorbeeld de 3.5 inch floppydrive, een 3.5 inch harddisk en 5.25 inch DOS- drive) is het drive-chassis al vol.
Een torenkast kan dit probleem voor de "power user" oplossen. Meer uitbreidingsruimte en een krachtiger voeding bieden alle mogelijkheden tot zwaar professioneel gebruik.

De tower

De benaming tower betekent in het Nederlands gewoon toren. Eigenlijk had men deze systeemkast beter flat of zoiets kunnen noemen, want daar lijkt deze vertikale systeemkast meer op. De term 'tower' geeft wel het idee dat er in de hoogte gewerkt wordt. Alle drives zijn boven elkaar gebouwd. De systeemkaarten staan dwars of vertikaal.



De Tower-case (=torensysteemkast)

Bij de IBM compatibele PC kent men de grote vloer-towers en de kleinere desktop-towers.

Big towers

De vloerbewoners of big towers horen met uitgeklapte pootjes (tegen het omvallen met "dag harddisk") onder of naast het bureau op de grond te staan. Dankzij de grote afmetingen kan de ware "freak" de systeemkast letterlijk volstouwen met hardware. Doorgaans gaat het om krachtige netwerksavers of krachtige (grafische/DTP-)werkstations.

Een computer op de vloer bespaart lekker op bureauruimte. Het enige nadeel is, dat de monitor wel over een grote in hoogte verstelbare draaivoet dient te beschikken, anders wordt het kijken lastig. N.B.: Normaliter staat de monitor op de systeemkast en u zult het hoogteverschil bij een tower moeten zien te overbruggen.
Een klein bijkomend probleempje met een vloer-tower is de lange kabelafstand van de systeemkast naar het bureau. Tal van monitor-, muis-, digitizer-, joystick- en andere kabeltjes blijken net te kort. En dat betekent weer in de beurs tasten voor verlengkabels!

Mini towers

De mini-towers of desktoptorens zijn eigenlijk gewoon vertikaal geplaatste kompakte AT-systeemkasten. Mini-towers strelen vaak het oog door de fraaie vormgeving. De ruimte is vaak te krap voor echt zwaar gebruik. Ook hier weer het probleem van de monitor- opstelling.

Power in Tower

De voornaamste reden om een Amiga 2000 of 2500 in een grote tower-behuizing te stoppen is pure uitbreidingskracht. Anders is dit wel een zeer dure manier om indruk op uw buurman computergebruiker te maken.
Voor wie komt een krachtige uitbreiding nu in aanmerking? We geven een aantal voorbeelden:

- Unix-gebruik. Bij de Amiga 2500 kunt u met Unix alle kanten op. Super-drives, 16 MB aan vrij RAM, uitgebreide netwerkconfiguraties enzovoort.
- Desktopvideo. Amateurs kunnen het met enkele externe randapparaatjes af. De (semi-)professional zit al gauw met een hele club randapparatuur: genlock, videoprocessor, RGB-splitter, HIRES non-flickering videokaart (Microway Flicker Fixer) en grote harddisks. Bij een tower past dat allemaal in één enkele behuizing.
- Grafische toepassingen. Door integratie van 33 MHz Motorola MC 68030-turbokaarten kan de Amiga zich met vele aanzienlijk duurdere grafische werkstation meten. CAD/CAM, animaties, ray-tracing-technieken, wetenschappelijke simulaties het kost allemaal een berg aan hardware en past alleen in een tower.
- Gebruik als een netwerk-saver

De towerkast voorziet, zoals gezegd, in de benodigde ruimte en voorziet in een krachtige voeding. Met name die voeding mag u niet vergeten, want al die extra hardware lust wel een slokje electronen en bij onvoldoende vermogen treden lastige storingen op.

Torens bouwen

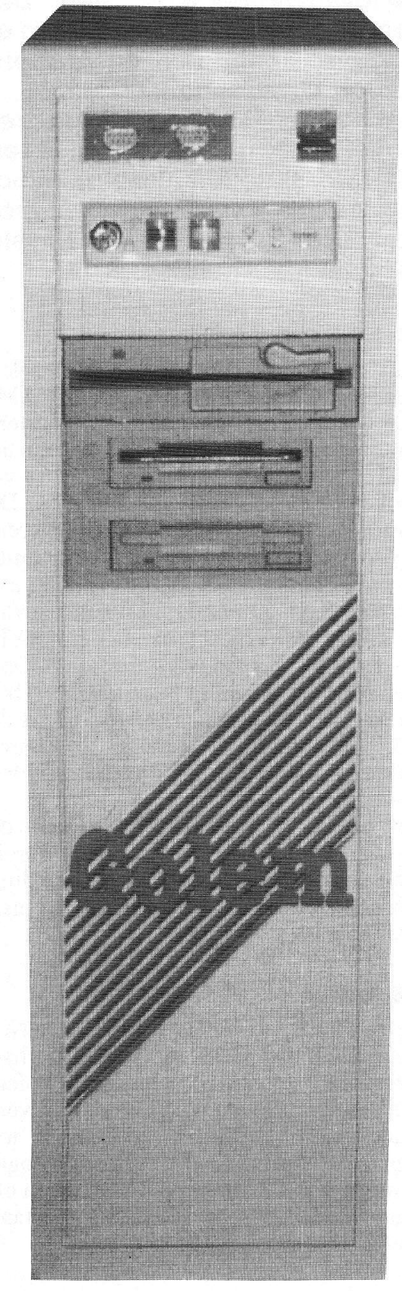
U kunt towers kant en klaar (tussen de *f* 8.000,- en *f* 15.000,-) kopen of zelf in elkaar zetten. Als eerste de zelfbouw.
Op de talrijke beursen en in advertenties van (duitse) postorderbedrijven worden losse torens met voedingseenheid tussen de *f* 400,- en *f* 1.000,- aan geboden. Iedereen die geen twee linker handen heeft kan hiermee zelf een tower-Amiga mee in elkaar knutselen.
Let bij aankoop wel even op de volgende zaken:

- De fysieke afmetingen en schroefgaten van de Amiga- hoofdkaart. Bij het vertikaal in de tower plaatsen van deze kaart in de towerkast moeten de gaatjes voor de bevestigingsschroeven boven de getapte bussen van de systeemkast vallen. Boren in printplaten is voor de leek een riskante aangelegenheid. Meet alles dus nauwkeurig op!
- Het vermogen van de voeding en het aantal beschikbare stroomkabeltjes. Hoe meer hoe beter.
- Hoeveel driveruimte zit er precies in de kast? Normaal is iets van 5-6 drives. Bijvoorbeeld drie 3.5 inch en drie 5.25 inch units.
- De mogelijkheid tot het plaatsen van uitbreidingskaarten. De meeste torens zijn voor PC-gebruik ontworpen. Dat is meestal geen fysiek nadeel, maar het blijft verstandig om de positie van connectoren, kaartvleugels en -geleiders toch even nauwkeurig op te meten.
- Het bedieningspaneeltje. Wat staat daar op? Bijvoorbeeld een RESET-knop, power-indicatie, harddiskindicatie. LEDs voor de kloksnelheid, aan/uitschakelaar en turbo-aanduiding.
- De degelijkheid van de Tower
- De kabelafstanden in relatie tot de connectoren

De grote voordelen van zelfbouw zijn de forse financiële besparing en het feit dat de Amiga-gebruiker zijn/haar systeem gefaseerd kan opbouwen.
In Engeland en Duitsland worden per advertentie diverse Amiga- towers aangeboden. Een aantal Amiga-specialisten (Golem, VideoComp, X-pert Computer Services) levert het systeem op maat aan de beter gefortuineerde cliënt. Er is keuze uit modellen met de Microway Flickerfixer, Multisync monitor, 33 MHz GVP 68030 turbokaart, laserdisks, harddisks van 100 MB of meer, GVP tapestreamers, vele MB's aan 32-bits, VGA-kaart en tal van floppy-diskdrives. Het is maar wat u er voor over heeft.
Ook kan er toepassingsgericht geleverd worden. Voor DTP-, DTV- en Unix zijn speciale configuraties leverbaar.

Power uit de Amiga-tower is een krachtige en vooral dure aangelegenheid. Wie niet zelf bouwt zal flink in de hobbybuidel moeten tasten. Voor de professional zal zoiets geen bezwaar zijn. Die heeft grif geld voor de geboden mogelijkheden en het geïntegreerde gemak over.

U.S.



CLS in Amiga Basic

Lees ik daar in Commodore-Info 89/7 (blz. 59) een hint/tip over een commando CLS, om het scherm schoon te maken en de cursor linksboven te plaatsen.

Het programmaatje is geschreven in "C".

Er staat nog wel bij: "Piepklein en wordt toch oh zo groot". Kryptisch, maar waar. Wat daarmee bedoeld wordt zal weldra blijken. Hieronder de source uit: Commodore-Info 89/7.

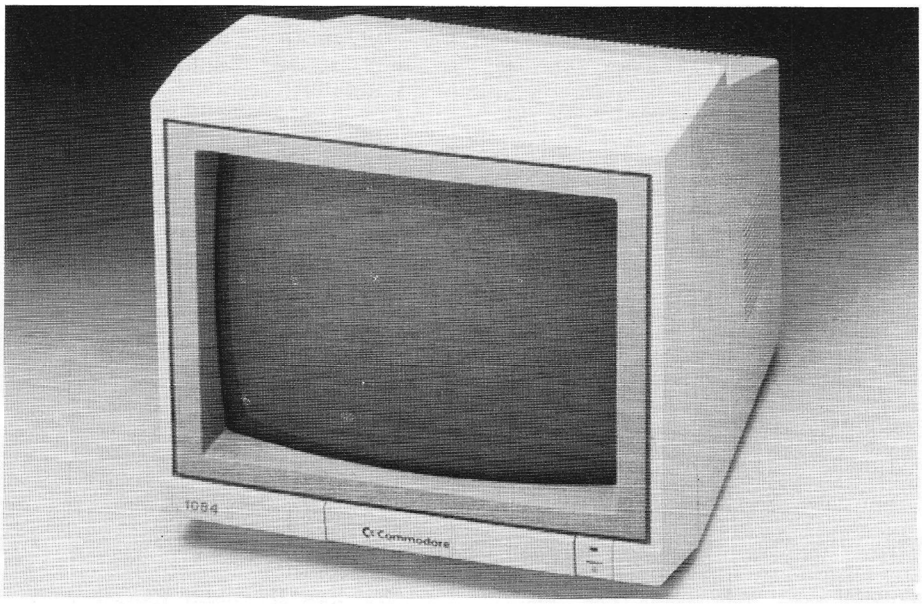
```
main ()
{
 printf("%c",12);
}
```

Veel te groot

Deze source "Cls.c" is inderdaad klein (30 bytes). De object-code "Cls.o" is al wat gegroeid (196 bytes). De uiteindelijke phase "Cls" blijkt echter reusachtig te worden (10.224 bytes). Maar ... het werkt perfect !!!
Ik heb gebruik gemaakt van Lattice-C 3.10.
Wel heb ik ergens gelezen, dat jongere versies van Lattice-C de mogelijkheid hebben om meer Amiga-gerichte en daardoor kleinere phases te fabriceren (minder mee te LINKen). Hoeveel kleiner staat er niet bij. Maar ik denk dat het voor het programmaatje "Cls" nog altijd veel te groot zal blijken.

Korter

Kan het niet veel korter ? Ja hoor, maar dan in "Assembler". Daar is niet iedereen erg handig in en ik zelf ook niet. Maar ik herinnerde mij een artikeltje in Commodore-Info 89/3 (blz. 95) over een klein programma, dat de tekst "Hallo wereld!" op het scherm tovert en dat programmaatje was geschreven in Assembler. Wel, dacht ik, als ik nu eens een nog korter tekstje naar het scherm stuur. Een tekstje van 1 byte lang, waarbij de inhoud van deze byte HEX-0C ($0C) is. Dan zou dat hetzelfde effect moeten hebben. Zo gezegd, zo gedaan. Hieronder de source uit Commodore-Info 89/3, welke door mij een beetje is aangepast na enig snuffelen in daarvoor bestemde boeken.



```
start:    move.l    ExecBase,a6
          lea       dosnaam,a1
          jsr       OpenLib(a6)
          move.l    d0,a6
          jsr       Output(a6)
          move.l    d0,d1
          move.l    #schoon,d2
          moveq     #1,d3
          jsr       Write(a6)
          clr       d0
end:      jsr       Exit(a6)
even
dosnaam: dc.b
    'dos.library',0
even
schoon:   dc.b      $0C

OpenLib  = -408
ExecBase =    4
Write    =  -48
Output   =  -60
Exit     = -144
```

Oplossing

Deze source "Cls.S" is 394 bytes lang. Ik heb hier gebruik gemaakt van SEKA-assembler 1.5. De uiteindelijk phase "Cls" is 136 bytes klein gebleven en het werkt perfect!
Leuk, aardig, schitterend ende hoera. Maar wat hebben al die Amiga-gebruikers er aan, die geen C-compiler hebben en ook niet in het bezit zijn van een Assembler ??? Njente, niets, noppes ende geen moer. Toch is er voor hen wel een oplossing. Wat hebben alle Amiga-gebruikers namelijk wel ? AmigaBasic !

Amiga Basic

Elk programma is in feite niets anders, dan een aaneengesloten rij LONGWORDs (groepje van 4 bytes) en bovendien is het gewoon een bestand en dus ook als zodanig te behandelen. En dat gaan we dan ook doen. We gaan eerst een bestand creeren, daarna vullen met de juiste verzameling LONGWORDS en vervolgens bewaren op diskette. Dit bestand kunnen we dan later gebruiken als programma of commando (U mag zelf kiezen hoe u het noemt). En dit alles doen we gewoon met AmigaBasic.

Hex.Load

Hieronder mijn AmigaBasic-oplossing, genaamd Hex.Load. Het is recht toe recht aan geschreven. Geen al te moeilijke constructies en zo min mogelijk commentaar. Zoals ik al zei, een programma bestaat uit een aaneengesloten rij LONGWORDs. De inhoud van elk LONGWORD wordt als element in een DATA-regel gezet, precies zoals het HEXadecimaal zou zijn geprint. Het handigste is 4 LONGWORDs in 1 DATA-regel. Dan ziet het er net uit als de HEX-dump van het programma.
Hex.Load leest de DATA-elementen,

maakt er weer fatsoenlijke LONGWORDs van en schrijft ze naar een bestand. Dit bestand is dan het betreffende programma en kan gewoon worden uitgevoerd.
Hieronder de listing van het AmigaBasic-programma: Hex.Load.

```
REM Dit programma heet
    Hex.Load
REM Geschreven door Axel
    Kohler
REM Versie van
    19-december-1989

REM Elke DATA-regel bevat de
    HEX-weergave van 1 of
    meer LONGWORDs

file$ = "df1:Cls"     'De naam
    van het te creeren
    programma
LONGWORDs% = 34       'Het
    aantal LONGWORDs in
    DATA-regels
checktotal% = 6904   'Checksum
    van het programma "Cls"

OPEN file$ AS #1 LEN=4
FIELD #1, 4 AS r$

checksum% = 0

PRINT "We zijn bezig, even
    geduld"
PRINT

FOR i% = 1 TO LONGWORDs%
     rr$ = ""
     READ a$
     FOR j% = 1 to 4
          fout% = 0
          h$ =
    MID$(a$,j%*2-1,1)
               GOSUB heks
               p1% = p% : h1$
    = h$
          h$ = MID$(a$,j%*2
    ,1)
               GOSUB heks
               p2% = p% : h2$
    = h$
          p% = p1% * 16 + p2%
          checksum% =
    checksum% + p%
          IF fout% = 1 THEN
    GOSUB foutje
          rr$ = rr$ + CHR$(p%)
     NEXT j%
     LSET r$ = rr$
     PUT #1,i%
NEXT i%

IF checktotal% = 0 THEN
   PRINT "De checksum is
    niet getest"
   END
END IF

IF checksum% = checktotal%
    THEN
   PRINT "De checksum is
    goed"
ELSE
   PRINT "De checksum is
    niet goed"
   PRINT "Controleer de
    DATA-elementen"
END IF

END

heks:
   p% = 99
   IF h$="0" THEN p%=0   :
    RETURN
   IF h$="1" THEN p%=1   :
    RETURN
   IF h$="2" THEN p%=2   :
    RETURN
   IF h$="3" THEN p%=3   :
    RETURN
   IF h$="4" THEN p%=4   :
    RETURN
   IF h$="5" THEN p%=5   :
    RETURN
   IF h$="6" THEN p%=6   :
    RETURN
   IF h$="7" THEN p%=7   :
    RETURN
   IF h$="8" THEN p%=8   :
    RETURN
   IF h$="9" THEN p%=9   :
    RETURN
   IF h$="A" THEN p%=10  :
    RETURN
   IF h$="B" THEN p%=11  :
    RETURN
   IF h$="C" THEN p%=12  :
    RETURN
   IF h$="D" THEN p%=13  :
    RETURN
   IF h$="E" THEN p%=14  :
    RETURN
   IF h$="F" THEN p%=15  :
    RETURN
   fout% = 1
RETURN

foutje:
   PRINT "Fout in
    DATA-element nr."; :
    PRINT i%;
   PRINT "    Byte nr."; :
    PRINT j%;
   PRINT "is: "; : PRINT h1$
    + h2$
   PRINT
   p% = 0
RETURN

DATA
    "000003F3","00000000","0000
    0002","00000000"
DATA
    "00000001","0000000F","0000
    0001","000003E9"
DATA
    "0000000F","2C790000","0004
    43F9","0000002A"
DATA
    "4EAEFE68","2C404EAE","FFC4
    2200","243C0000"
DATA
    "00367601","4EAEFFD0","4240
    4EAE","FF70646F"
DATA
    "732E6C69","62726172","7900
    0C69","62726172"
DATA
    "000003EC","00000002","0000
    0000","00000008"
DATA
    "0000001A","00000000","0000
    03F2","000003EB"
DATA "00000001","000003F2"
```

HEX-dump

Uiteraard kun je Hex.Load gebruiken om elke combinatie van LONGWORDs in een bestand te zetten. Op die manier kun je elk programma zelf op diskette zetten en daarna gewoon gebruiken. Hiervoor moet je wel een print hebben van een HEX-dump van het gewenste programma.
Echt bruikbaar is deze methode alleen als het betreffende programma niet te lang is. Maar voor de echte doordouwers is de enige beperking het voor AmigaBasic beschikbare geheugen en dat kan nog worden vergroot via het AmigaBasic-commando CLEAR.

WHAMP

Stel, dat u een print hebt van een HEX-dump van een commando met de naam: WHAMP. Hoe kunt u dan het programma Hex.Load gebruiken om WHAMP op diskette te krijgen ?
U dient dan voor het volgende te zorgen:

- Verwijder de oude DATA-regels (van Cls).
- Voeg de nieuwe DATA-regels toe (van WHAMP).
- Geef de variabele file$ de waarde df1:WHAMP of ram:WHAMP.
- Tel het aantal LONGWORDs van WHAMP (u hebt een printje).
- Geef de variabele LONGWORDs% de hierboven gevonden waarde.
- Bepaal de checksum van WHAMP. Voor een byte met inhoud $FA is de bijdrage aan de checksum 15*16 + 10 = 250.

Geef de variabele checktotal% de hierboven gevonden waarde. Als u checksum niet hebt willen/kunnen bepalen, zet dan 0 (nul) in checktotal%. De checksum wordt dan niet getest.

R.C.A.B. Kohler
Gerrit van der Veenstraat 37
3762 XH Soest
Tel: 02155 - 13487

68030 Turbokaarten

Veel power voor een hoge prijs

Voor de Amiga-gebruiker die een grafisch werkstation ambieert, is een MC 68030- een relatief voordelige uitkomst. Een kaart in een leeg Amiga-uitbreidingsslot pluggen en het systeem draait 10 tot 12 maal sneller. Prima voor raytracing-technieken, (wetenschappelijke) simulaties en flitsende animaties. De aanschaf zal voor menigeen echter een forse aanslag op het hobbybudget betekenen.

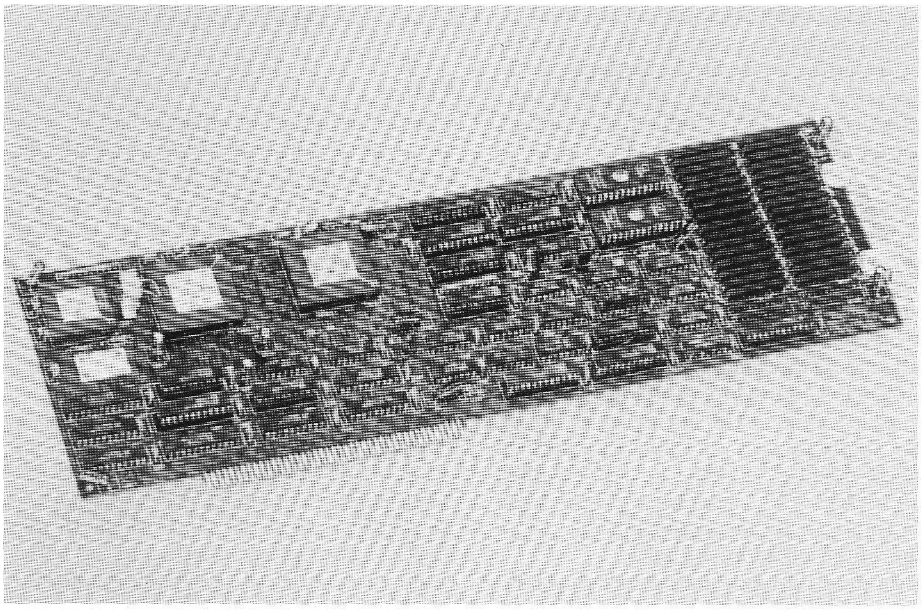
Goud

De oude 7,14 MHz MC 68000 CPU doet het nog best en kan zeker in de AT-klasse best meekomen. Wie meer wou en over voldoende slappe was beschikte kocht wellicht al een Amiga 68020-model. Voor sommige toepassing gaat deze CPU-kracht nog niet ver genoeg. In de professionele wereld van real-life graphics, simulaties en animaties is elk pietsje extra hardwarekracht goud waard. Wie niet wil wachten tot hij/zij een ons weegt zal in een MC 68030 turbokaart moeten investeren.

MC 68030

Nieuw is duur. Een bekend feit uit de computerwereld. Weet u nog de introductieprijs van de Amiga 1000? Die lag in Nederland rond de *f* 7.000,-. Nu betaalt de liefhebber een zelfde bedrag voor de Amiga 2000 met 14,3 MHz Motorola 68020. Daar zit veel kracht in zo'n machientje en Commodore schermt dan ook levendig met professionele DTP-, Unix- en grafische toepassingen.

De gewone man met een krap hobbybudget kan tegenwoordig voor tussen de *f* 1.500,- en *f* 2.000,- een MC 68020-turbokaart kopen; dat kost één uitbreidingsslot. Natuurlijk heb je dan nog geen echte Amiga 2500 want daar horen minimaal 2 MB aan vrij RAM, mathematische coprocessor en een zware harddisk in. Het begin is er echter en de nijvere gefaseerde doorbouwer zal het nog ver schoppen.



Voor wie 14,3 MHz nog niet hard genoeg gaat, is er de MC 68030- optie. Zowel Commodore als enkele onafhankelijke hardwareleveranciers bieden deze kaarten tegen een stevige prijs aan. Bij die basisprijs komen dikwijls ook nog een mathematische coprocessor en extra geheugen (op kaarten met een 32-bits bus) om optimaal van de geboden CPU-kracht te kunnen profiteren.

68030-kracht

Het hart van de 68030-turbokaarten wordt gevormd door de gelijknamige 32-bits Motorola microprocessor. Tot de specificaties van dit rekenwondertje horen o.a.:

- 32-bits geheugenbus (= 32-bits Non-Multiplexed Address- en Data-bus). De Dynamic Bus voor naar keuze 8-bits, 16-bits of 32-bits geheugenbanken en randapparatuur;
- 4 Gigabyte aan direkte adresseerruimte;
- 16 registers van 32-bit;
- 256 byte on-chip instruction cache;
- 256 byte on-chip data cache;
- on-chip Memory Management (PMMU);
- Pijplijnarchitectuur met mogelijkheid tot parallel processing;
- Kloksnelheid van 16, 20, 25 of 33 MHz;
- Compatibel met de complete 68000-familie van CPU's en mathematische coprocessoren;
- HCMOS-technologie;
- DRAM burstmodus voor vier datawoorden (DMA);
- Synchrone en asynchrone buscommunicatie;

Sommige kenners vinden de MC 68030 gewoon een upgrade van de MC 68020. Anderen spreken toch van een nieuw CPU-ontwerp. Onafhankelijk van wie er nu gelijk heeft, blijft de MC 68030 een van de krachtigste PC-chips die momenteel verkrijgbaar is.

De kloksnelheid wordt in de praktijk op 16 of 25 MHz gezet, waarbij de snellere uitvoering al snel 2 tot 3 maal zo duur uitpakt.

In zijn eentje komt de MC 68030 CPU zoals gezegd niet echt snel uit de voeten. Met behulp van de mathematische MC 68882 coprocessor (20 tot 33 MHz) nemen de prestaties flink toe. Doe er dan ook nog wat dure SRAM MegaByte-banken bij, en er staat een snelheidsspetter van een computer op het bureau. Hoogwaardige grafische technieken, Unix, professionele DTP en DTV, alles loopt als een supersnelle trein op de Amiga met 68030 turbo-board.

De keuze

Voor een "dubbeltje" op de MC 68030-snelheidsrang kan met het Professional-030-Board van Harms Computersysteme. (Leverbaar via Kupke en andere postorderbedrijven). Dat "dubbeltje bedraagt overigens voor de Amiga 500 en 2000 zo'n *f* 2.200,- en daar komt de dochtergeheugenkaart nog bij.
De Profesional-030-kaart versnelt de grafische prestaties van de Amiga met minimaal 200%. Meer kan, maar dan moeten hard- en software wel meehelpen. Het is de bedoeling om de kaart optioneel met 4 MB aan 32-bits Fast-RAM te completeren. De installatie moet softwarematig gebeuren. Dankzij de asynchrone timing kan de hardware met elk beschikbare klokfrequentie werken, terwijl de oude MC 68000 CPU gewoon op 7,14 MHz doorpruttelt. Indien nodig kan de turbokaart uitgeschakeld worden.
In de praktijk biedt de Professional-030-Board redelijke prestaties voor zijn geld. Problemen kunnen zich voordoen bij tegen kopiëren beschermde software.

Great Valley Products (GVP) staat al jaren bekend als leverancier van stevig geprijsde klasse randapparatuur voor de Commodore Amiga-lijn. Halverwege 1989 kwam de Impact A2000-030 turbokaart uit. Van deze MC 68030 turbokaart vliegen de snelheidsvonken af. Menig Unix grafisch werkstation kan aan deze 1200% versnelling ten opzichte van de 7,14 MHz MC 68000 CPU nog een puntje zuigen. Aan deze 25 MHz MC 68030-plus MC 68882 hangt helaas een prijskaartje van tussen de *f* 6.000,- en *f* 7.000,-. Bij deze hoge prijs zijn wel een harddiskcontroller en 4 MB aan kwaliteits Fast-RAM inbegrepen.

Unix

Unix komt binnen het Amiga-bereik door in de desbetreffende ROM- sockets de Unix boot-ROMS te steken. Dan kunt u de Amiga als grafisch Unix-werkstation of als netwerkcomputer inzetten. Dankzij de voorzieningen in Amiga DOS 1.3 kan vanuit dit besturingssysteem met Unix gestart worden. Desgewenst kan de gebruiker ook rechtstreeks onder Unix opstarten. Amiga professional in optima forma.
Natuurlijk blijft het ook mogelijk om de Impact turbokaart uit te schakelen en weer op de oorspronkelijke MC 68000 CPU verder te draaien.
Een zeer goede 68030 turbokaart die dankzij asynchrone bursts op 25 MHz (binnenkort zelfs op 33 MHz) gigantische prestaties weet te bereiken. Als enige bezwaar geldt de aankoopprijs die alleen de professional kan opbrengen.

Commodore zelf betreedt met de 25 MHz A 2630 de 68030 turbo-arena. Voor rond de *f* 6.000,- koopt u een 25 MHz MC 68030, MC 68882 mathematische coprocessor en 4 MB aan custom-RAM. Voor RAM- uitbreiding zijn speciale RAM-expansiekaartjes nodig.
Ook deze turbokaart mept de Amiga 2000-prestaties met een factor 10 omhoog en brengt via de ROMS Unix op het Amiga-bureau.
Zondermeer een prima turbokaart die in de praktijk iets minder snel is dan zijn GVP-collega en wat lastiger methode van RAM- uitbreiding hanteert.

MC 68040

MC 68030-turbokaarten liggen eigenlijk nog niet binnen het bereik van de hobbyist. Zij bewegen zich in het professionele, Unix- en High-Tech-bereik van de Amiga. Alleen de Proffesional-030-Board maakt voor relatief weinig geldt een grafisch werkstation van uw Amiga. Dit echter zonder Unix. In de nabije toekomst zullen de prijzen ongetwijfeld dalen, maar dan is vast de nog snellere MC 68040 turbokaart al uitgekomen. Let altijd wel goed op de compatibiliteit. Eerst de eigen Amiga-uitvoering bekijken, dan de leverancier naar de specificaties vragen en vervolgens pas kopen.

Inlichtingen: 040-417596.

U.S.

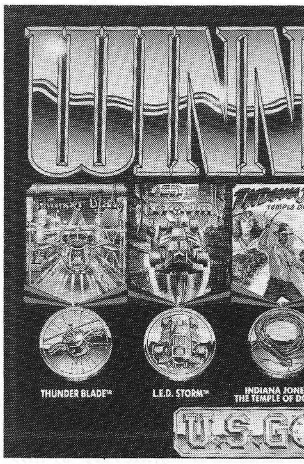
Oud van Goud

Ditmaal een aantal Gouwe Ouwe in een nieuw jasje. Nu eens niet diep in de kast en onder het stof een aantal pakketten, maar splinternieuwe oude software zo uit de verpakking en toch 'oud'

EPYX is een fabrikant die al vele jaren op de Commodore-markt actief is. De naam EPYX staat dan ook garant voor kwaliteit en vele uren spel plezier. Nu waren deze spelen in het begin dan ook niet altijd de goedkoopste software. Het doet ons dan ook een groot plezier, dat EPYX de laatste tijd deze software weer opnieuw uitbrengt. En nu niet per spel, nee, een doos met maar liefst vijf diskette's erin. De laatste paar maanden zijn er daarvan twee op de markt gebracht. Voor de rekenaars onder ons: het zijn geen 10 verschillende spelen. Op de een of andere manier, waarschijnlijk omdat het erg goed spel is, zit er in beide dozen het spel Impossible Mission II. Dus 9 maal spel-plezier in twee dozen voor de prijs van 1 goed spel. Het kan haast niet beter. We zullen in het kort enkele van deze spelen bekijken.

California games.

Dit spel is een logisch vervolg op de rage van Summergames, Wintergames, Ughgames enz. Een groot aantal sporten passeren ook hier weer de revu. Natuurlijk staat dit spel al garant voor vele avonden computerspel. Een van de sporten hierbij is het Half-pipe skate boarding. De bedoeling hierbij is natuurlijk om zoveel mogelijk punten te behalen in een halve buis. Hoe gewaagder de stunt, hoe hoger de waardering wordt. Vallen levert helemaal geen punten op, dus oppassen blijft de boodschap. Het is, zeker in het begin, al een hele kunst om alleen maar te blijven staan. Footbag, is een onderdeel waar een klein balletje zo lang mogelijk omhoog moet worden gehouden. Dit mag met het hoofd als ook met de voet gebeuren. Wat five in a row, dizzy dean, half axle en een head-bang betekenen, komt u snel genoeg te weten, het is namelijk onmisbaar om dit onderdeel te kunnen winnen. Surfing, bij de meeste mensen wel bekend, alleen gebeurt het hier op de Amerikaanse manier, dus boven op de golven. Om hoog te kunnen scoren, moeten er boven op de woeste branding de meest ingewikkelde figuren worden uitgevoerd. Bij het roller-scating moet er langs het strand op de boulevard een parcours worden afgelegd. Ook hier behoren extra punten tot de mogelijkheden maar kapotte knieën zijn hier meestal het resultaat als je te overmoedig wordt. De echte halsbrekende toeren zijn uit te halen bij het BMX-bike racing. Een heuvelachtig parcours met een groot aantal springschansen ligt te wachten op de deelnemers. Probeer je tegenstanders voor te blijven, de enigste remedie om hier te winnen. Frisbee, U weet wel van de vakantie, het gooien met zo'n bordje. Hoe nauwkeuriger er gegooid wordt, hoe hoger het aantal punten zal zijn.



Wintergames

Het kon bij zo'n verzameling niet uitblijven: de oude vertrouwde Wintergames zijn weer terug. Met dit spel is het eigenlijk allemaal begonnen. We bevinden ons hier op een winters strijdperk. Bobsleënd, met een duizelingwekkende vaart, moet het parcours worden afgelegd. Moeilijk, omdat er twee delen van het scherm in de gaten moet worden gehouden, om heelhuids beneden te komen. Bij het figuurschaatsen kunnen, mits je handig en zeer snel bent, de meest fantastische figuren worden gemaakt. Hoe mooier, hoe hoger, des te meer punten zijn er te behalen. Speedskieën gebeurt via een duidelijke aanslag op de joystick. Neem hier niet een te lange afstand, want een joystick geeft het eerder op dan je denkt. De berg af bij het slalomskieën, is hierna een ware verademing, op tijd sturen en je kunt een heel eind komen zonder al te grote inspanning. Het skispringen behoeft, mits er niet te moeilijke sprongen gemaakt worden, geen al te grote moeilijkheden op te leveren. Down Hill, letterlijk 'van de heuvel af', is niets anders dan een vrij val in de witte wereld. Het is veel moeilijker dan het lijkt. Wintergames wordt netjes afgewerkt met de nodige muziek en alle andere welkoms- en sluitings ceremonies, net zoals het geheel in werkelijkheid wordt omlijst.

4x4 off the road

De ware race-liefhebber komt bij dit spel zeker aan zijn trekken. Je hebt een keus uit een viertal voertuigen, storm troopers from Cox motors, de tarantula by venerable Motors, The Highlander by Lorry en als laatste Kantana from Oyama Motors. Nu zegt het u waarschijnlijk nog niets, later des te meer. De tweede keuze die moet worden gemaakt is het terrein waarin gereden moet worden. Er is een modderachtig, een winter maar ook een heuvelachtig parcours. Welke auto in welk parcours het beste voldoet, moet U zelf maar uitvinden; elke auto heeft zijn specialisatie. Voordat er gestart gaat worden, moet er eerst worden gestopt in de werkplaats. Hier is de auto uit te rusten met de onderdelen die u denkt nodig te hebben op het wedstrijd terrein. Is dit gedaan, dan niet vergeten langs de garage te rijden om de nodige reserve-onderdelen en olie te ha-

len. Nu is het punt bereikt om van start te gaan. Pas onderweg op: er bevinden zich veel obstakels, vaak op het allerlaatste ogenblik pas zichtbaar. De snelheid is niet altijd alles, soms gaat het beter als de snelheid wordt gematigd.

Basketball

Officieel heet dit spel Street sports basketball. Met dit spel is het mogelijk om op je luie stoel een spelletje basketball te spelen. De speldiskette en een Commodore zijn voldoende om te kunnen spelen. Er kan een keus worden gemaakt uit verschillende versies van het basketball spel, een spelletje op het schoolplein, in een steegje of op een parkeerplaats. Alles heeft zijn specifieke eigenschappen. Het is mogelijk om je eigen medestanders te kiezen. De handleiding, helaas in het Engels, is hierbij een noodzakelijk kwaad. Voor het samenstellen van je team staan er bekende spelers te trappelen om mee te mogen spelen, Butch, Radar, Jullie, Ralph, Magic en Kevin om er maar een paar te noemen. Ieder heeft zijn eigen specialisatie, dus ervaring kan je helpen winnen. Er kan zowel tegen de computer als tegen een tegenstander worden gespeeld.

Impossible Mission II

De naam geeft het eigenlijk al aan, een haast onmogelijke opdracht staat je te wachten. Je hebt na het opstarten van dit spel acht uur de tijd om de wereld van de ondergang te redden. Dit gebeurt door binnen deze tijd de juiste computer met de bijbehorende code te vinden. Het zal blijken dat je de acht uur hard nodig hebt, want er staan heel wat personages klaar om je te weerhouden de opdracht te vervullen. Zodra het spel wordt opgestart verschijnt er een informatie-paneel op het scherm. Hierop is van alles af te lezen. Het gehele spel speelt zich af in een achttal torens. Hier is te zien in welke toren je je op dat moment bevindt. In het midden bevinden zich drie torens; gestart wordt in de middelste toren. Aan elke liftschacht bevinden zich links en rechts een aantal kamers. Om deze kamers draait het allemaal. Op de rechterzijde van het informatiescherm zie je een computer. Hierop worden de muziekstukken bewaard die in de verschillende kamers verstopt zijn. Deze stukjes muziek moeten op de juiste manier achter elkaar worden gezet, hiervoor wordt het bedieningspaneel van de computer gebruikt. Elk muziekstukje mag slechts één keer voorkomen. Kamers waar je bent geweest worden zwart gekleurd. Laten we eens kijken wat er in de kamers te vinden is.
Het eerste wat opvalt zijn de vijanden, al zijn ze niet altijd als zodanig te herkennen. We kunnen alleen vertellen, ze zijn talrijk en lastig. Er zijn zeven verschillende types. Een aantal bezitten gevaarlijke laserstralen, deze zijn altijd dodelijk. Zij beginnen te spuiten op het moment dat je bij ze in de buurt komt. Een leven is hierbij niet te verliezen, maar het kost elke keer tien kostbare minuten van de kostbare speeltijd. Er zijn ook robots die mijnen leggen, maar met een beetje oefening zijn deze mijnen eenvoudig te ontwijken. Er zijn een aantal die geen kwaad doen, maar het je erg lastig kunnen maken door op een platform te gaan staan. Hierdoor kom je nog wel eens op de verkeerde etage terecht. Pas op voor buldozers, deze duwen je weg, je wordt verpletterd tegen een muur of je stort dan in een afgrond, een ijzingwekkende gil en weer zijn er tien minuten verloren.
De springveren zijn geen echte hindernissen, met wat oefening zijn ze zelfs als hulpmiddel gebruiken. Zij staan stil, tenzij je er op gaat staan. Verder tref je in de kamers hier en daar voorwerpen aan, dit verschilt per toren. Elke keer dat het spel wordt opgestart, zijn de voorwerpen op een andere plaats, ja zelfs in andere kamers. De bedoeling is om voor elk voorwerp te gaan staan en de joystick naar voren te bewegen. Er komt dan een afbeelding naar voren, een cijfer, een symbool of als je pech hebt niets. Cijfers zijn er in drie kleuren en dienen om een code te krijgen. De centrale computer bewijst ook hier weer zijn nut. Met de pijlen kan er een combinatie worden gevormd. Heb je de juiste combinatie gevonden, dan wordt de muur geopend die tot dan toe gesloten geopend en is de volgende toren te onderzoeken. Maar voordat je de toren verlaat moet eerst de brandkast, die zich in één van de kamers bevindt gekraakt worden. Hiervoor is het BOM-symbool nodig. Dit gebeurt door het BOM-symbool te kiezen uit het voorraadscherm. Is dit niet aanwezig dan moet deze eerst gezocht worden. Alle symbolen op dit scherm hebben een betekenis. Linksboven staat het platformsymbool, hiermee is een lift terug te halen. Bevriezers, deze zijn heel nuttig, robots kunnen hiermee tijdelijk bevroren worden. Rechtsboven staat de al eerder vermelde bom. Daaronder het mijn symbool, een tegenstander kan door aanraking ontploffen, maar pas op, wie een kuil ..... Het lampjes symbool heb je nodig als je in een donkere kamer komt om het licht aan te doen. De brandkast wordt opgeblazen door een bom ervoor te leggen, en er wordt een flink gat geslagen in de kast. Nu is het zaak voor dit gat te gaan staan en je vindt de microfoon. Er wordt nu automatisch een melodie opgeslagen op de centrale computer. Verlaat de kamer en controleer of de melodie werkelijk is opgeslagen. Zodra je meerdere schachten hebt doorzocht, moet je goed opletten. Er mogen geen tweemaal dezelfde melodieën worden bewaard. Is dit wel het geval, dan terug spoelen naar het begin van de dubbele melodie. Nadat je op deze manier zes verschillende melodieën hebt opgezocht, staat de bandteller op 150. Nu kun je naar de hoofdkontrole kamer gaan. Die gebeurt met de geheime lift, die alleen open gaat met zes verschillende melodieën. Deze kamer heeft een drietal computers, één van deze computers is echter de juiste en let op de tijd. Is die nog voldoende om voorzichtig door te gaan of moet er risico worden genomen? Is het je gelukt om de wereld te redden? Proberen maar!



Indiana Jones

Indiana Jones is een Arcade achtig aktiespel. Het spel beschikt over 3 levels, waarbij acht kinderen moeten worden bevrijd, die gevangen zijn genomen. De grote boosdoener hierbij is Thugee. Het is een warboel van ladders, kleine doorgangen, bewakers, watervallen en transportbanden. De kinderen zijn diep in het uitgebreide grottenstelsel opgesloten. In het tweede level staat een karretje centraal. Moeilijk wordt het met de opgebroken rails; ook hiervoor is Thugee verantwoordelijk. Lukt het je om het laatste level te komen, dan kom je in het mooiste gedeelte van het spel. Het strijdperk is hier de Temple of Doom. Indiana moet een belangrijke steen zien te vinden, de Sankara Stone. Deze is geplaatst voor het Kali beeld. Kali is een vierarmig beeld en staat bekend als de god van de Dood. Dit zegt al genoeg. Lukt dit toch dan moet er nog maar één hindernis worden genomen, en wat voor één. Een wankele touwbrug...